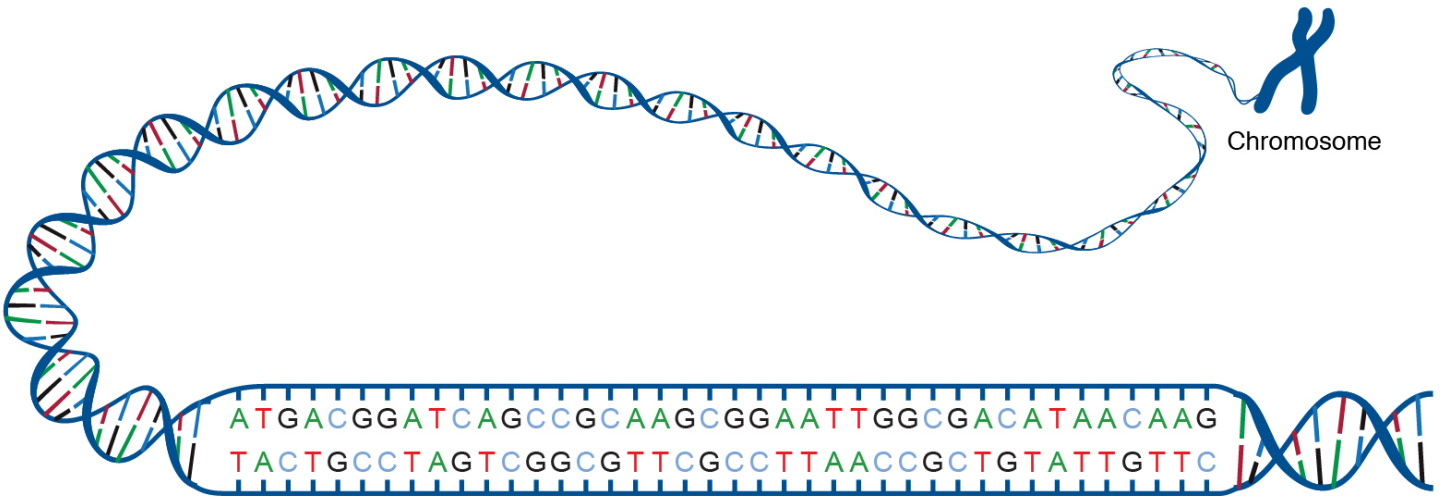
A little tour of assembly methods

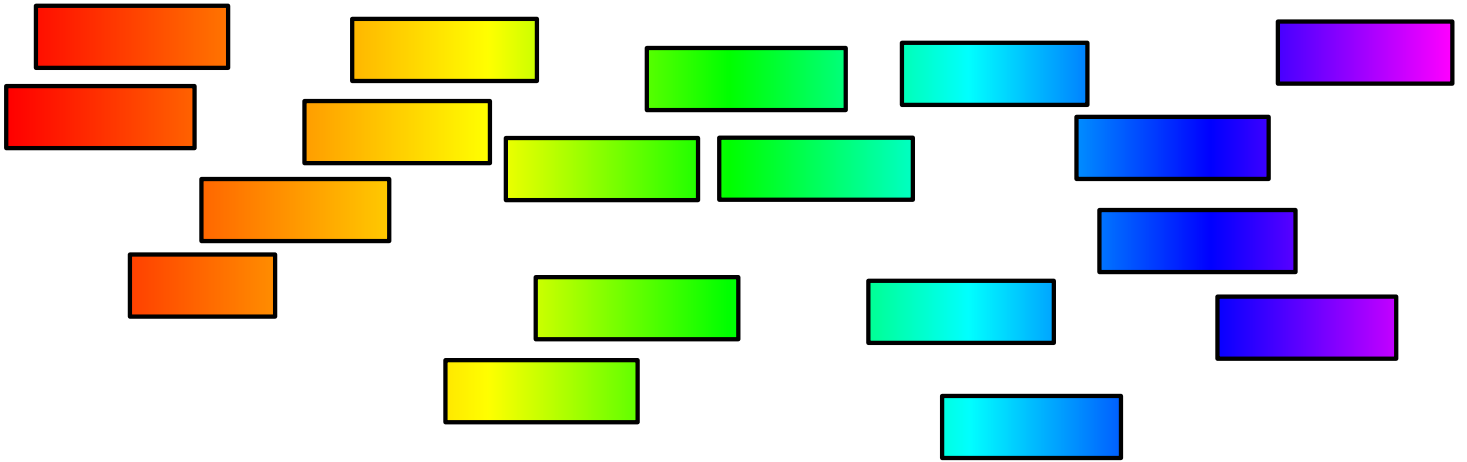
Antoine Limasset Bonsai, CRIStAL, Université de Lille,
CNRS CDP “PIE” Kick-Off Meeting


Na Ostrově

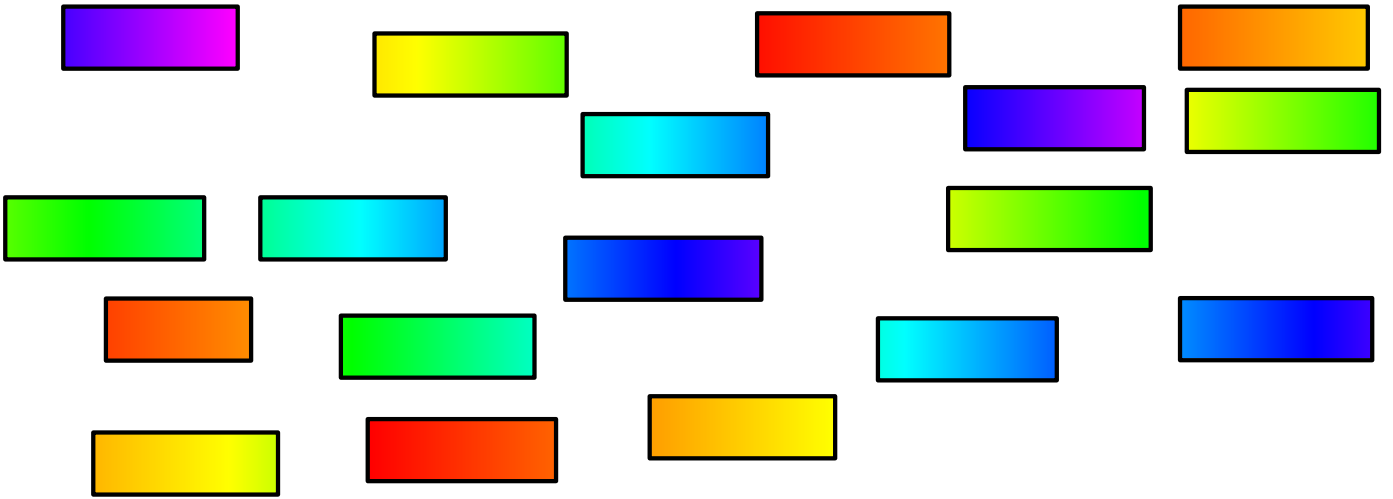
• Accessing a genome



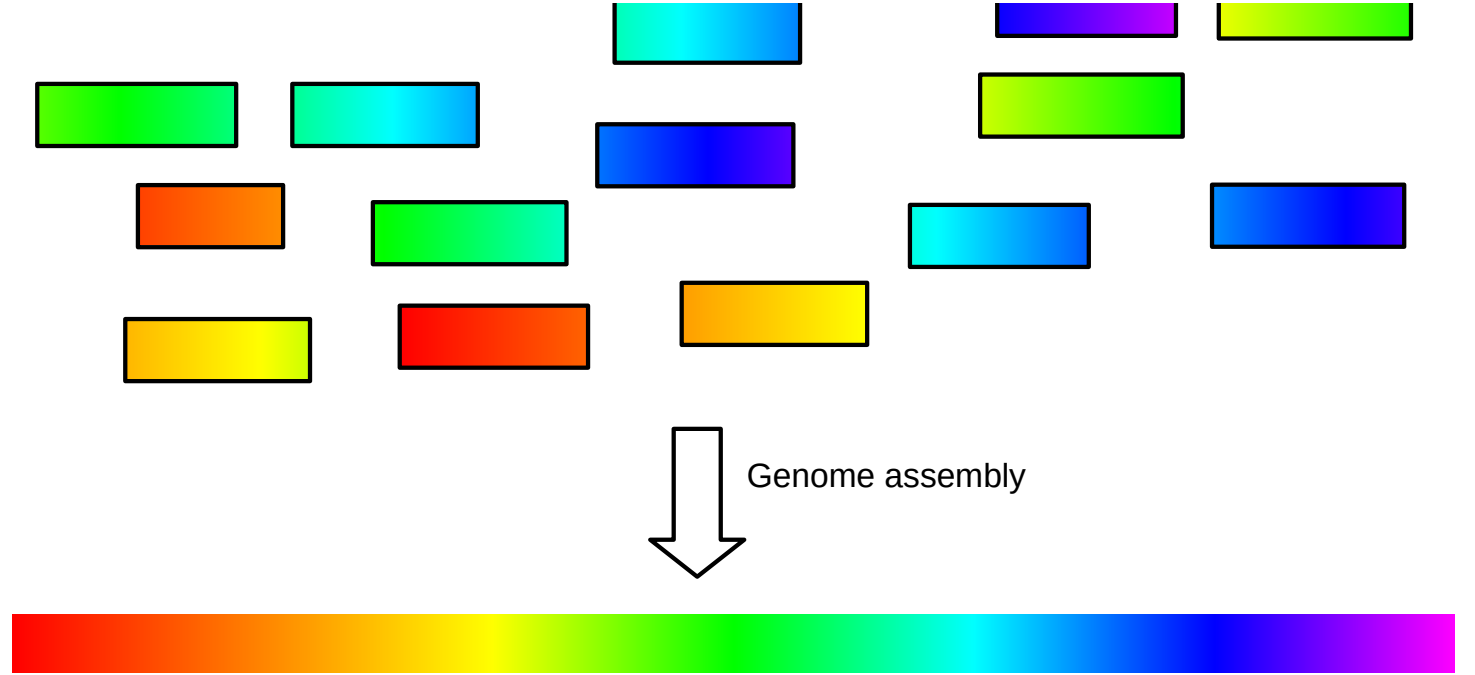
- **Reads are subsequences from the genome**



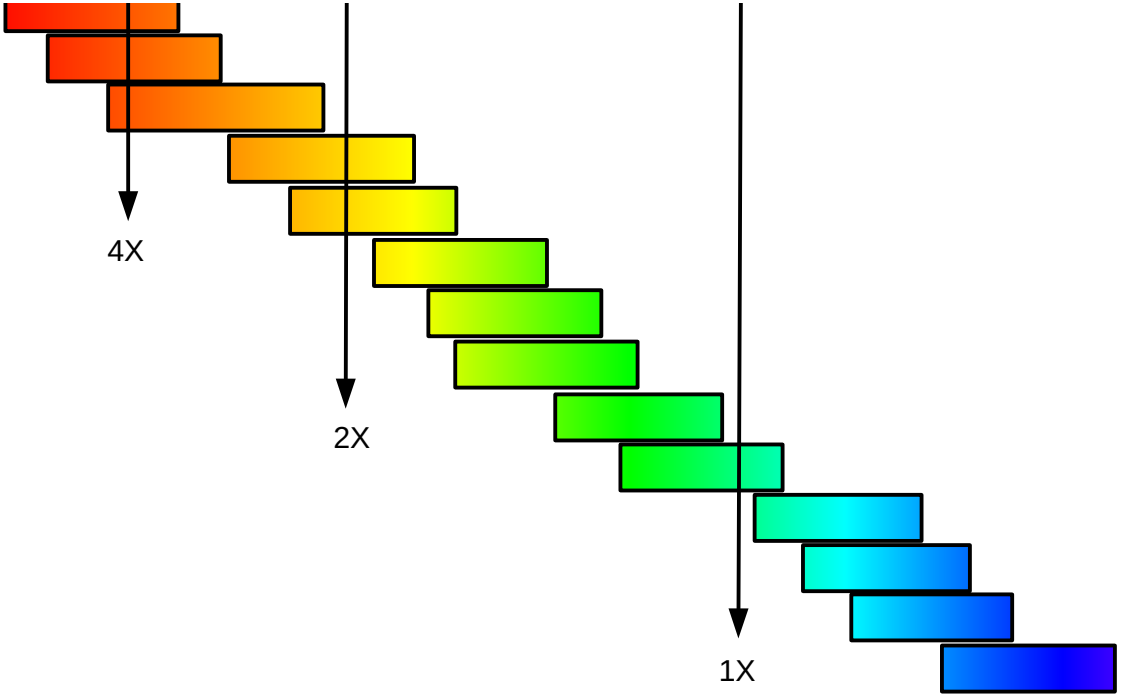
- **Reads are shuffled subsequences from the genome**



• Genome assembly task

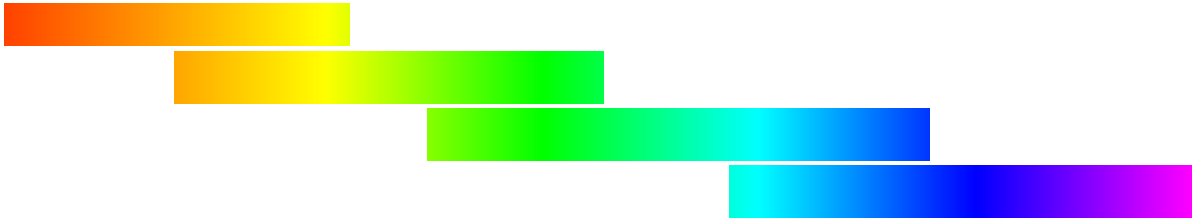


Genome sequencing: coverage & depth

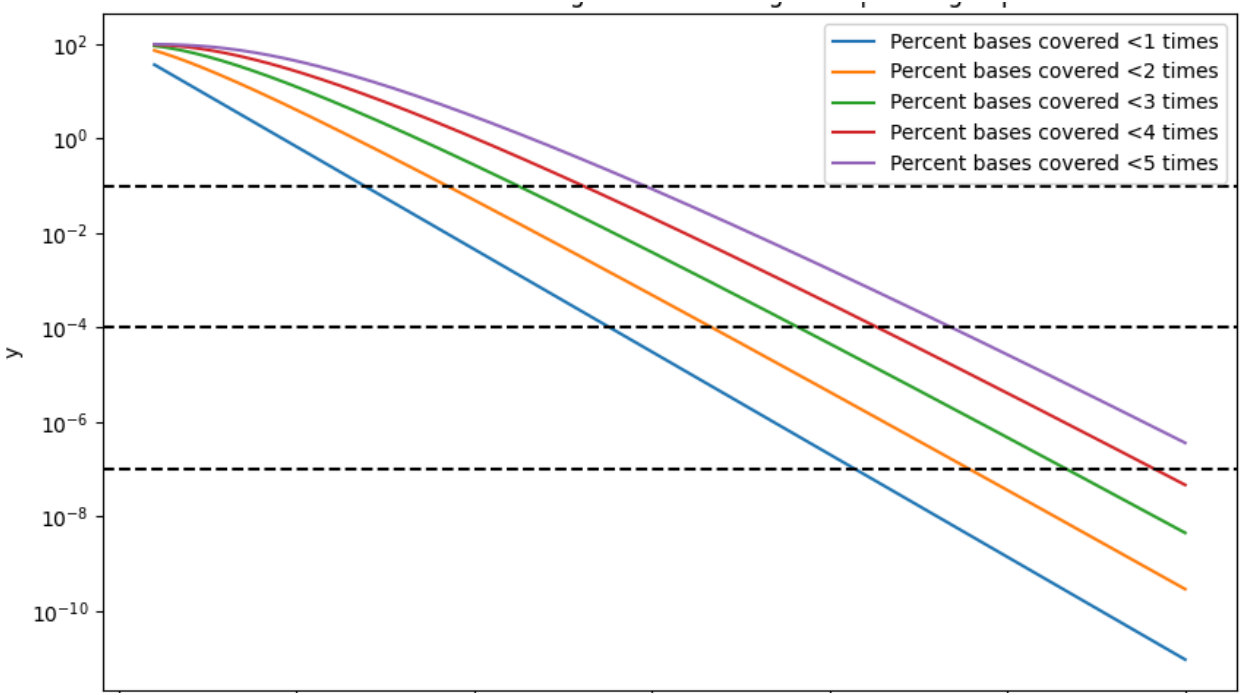


Genome sequencing: coverage

Genome sequencing: coverage



Poisson law


Percent bases covered <1 times
Percent bases covered <2 times
Percent bases covered <3 times
Percent bases covered <4 times
Percent bases covered <5 times
y
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}

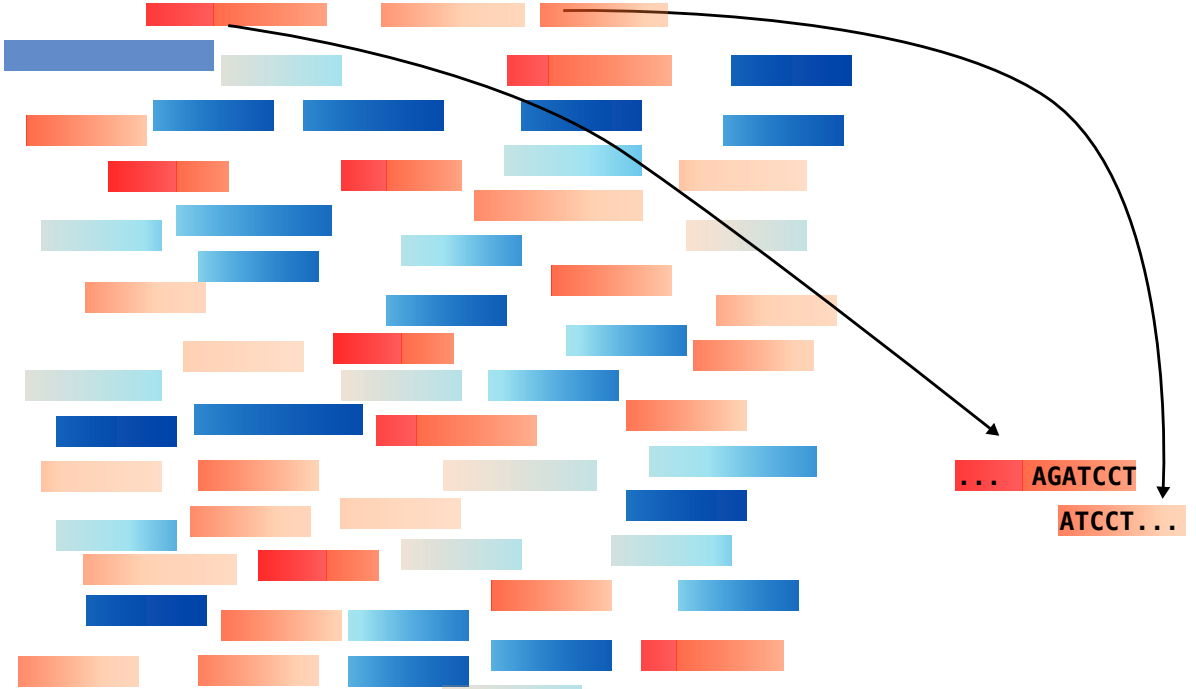
Poisson law (ii)

30-60X are often required for assembly projects

Longest overlaps

We are more confident in longer overlaps

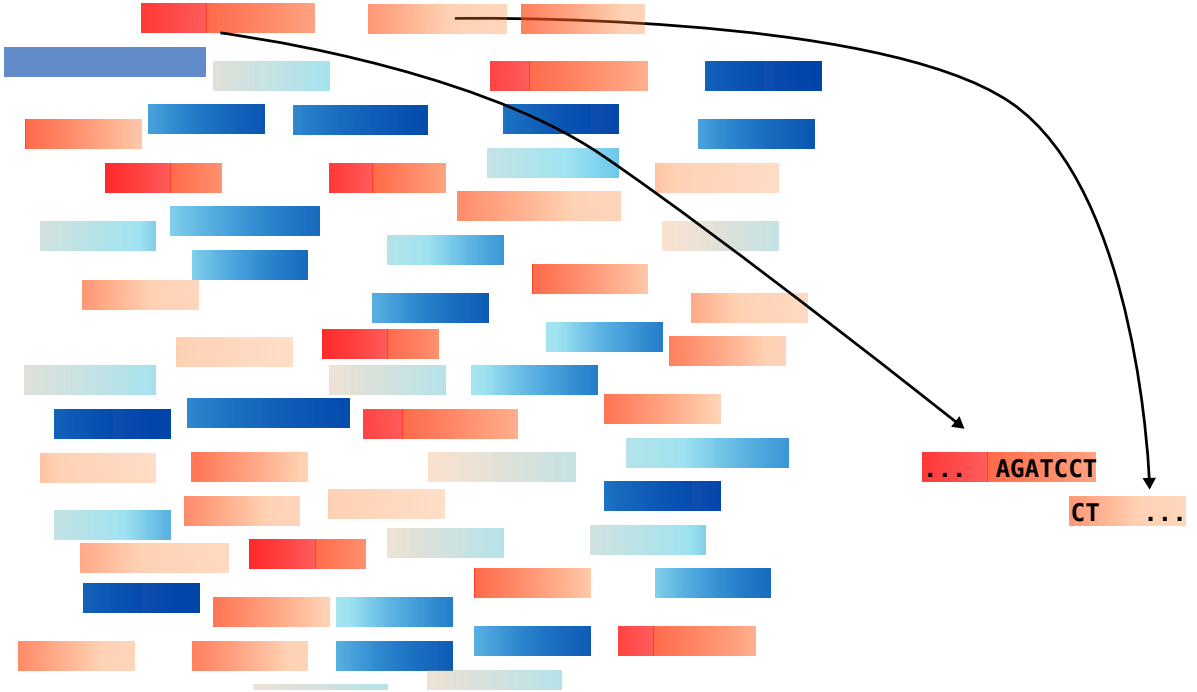
• Longest overlaps (ii)



• Small overlaps

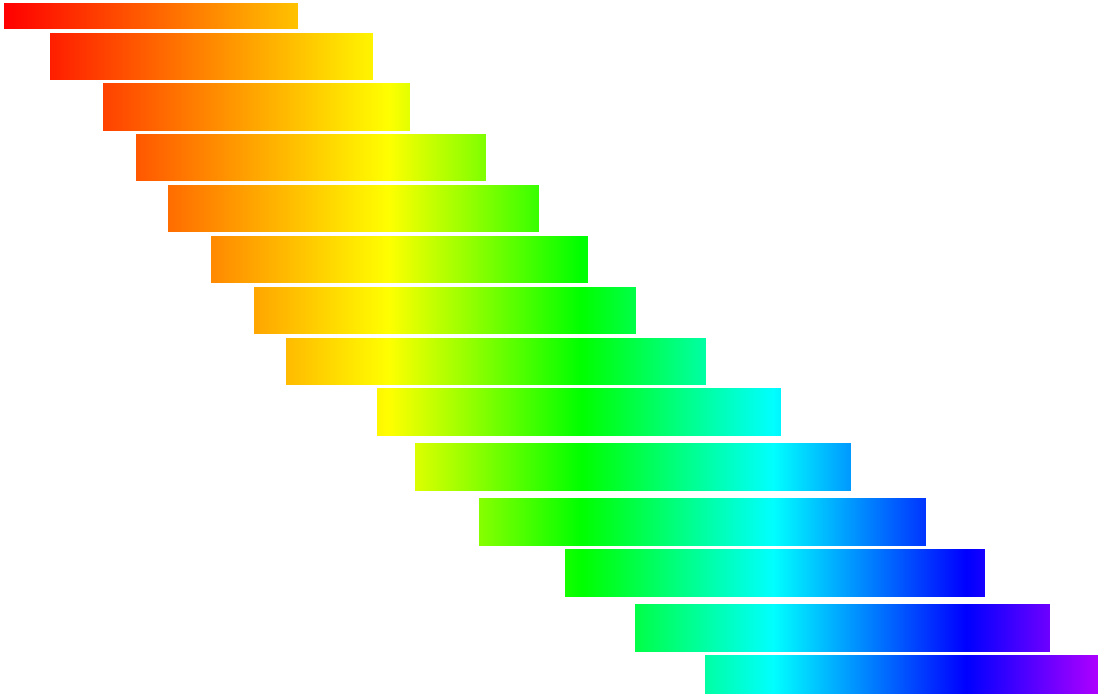
Can happen “by chance”

• Small overlaps (ii)

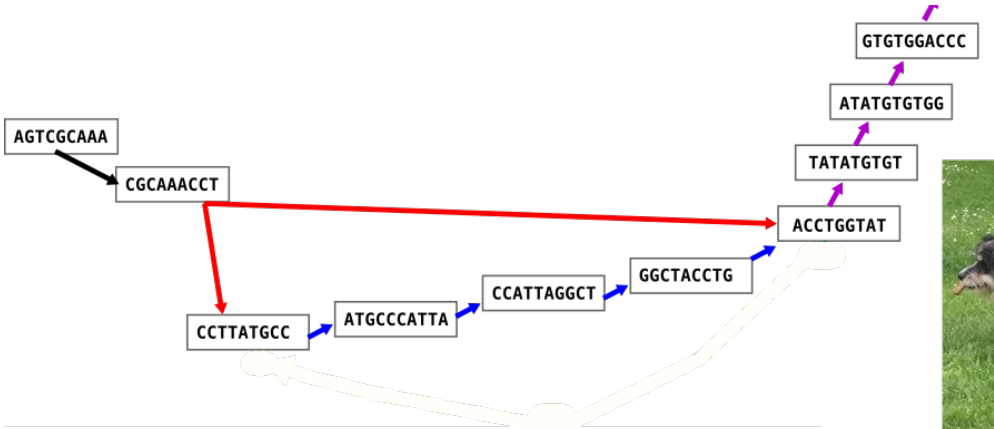


Back to coverage

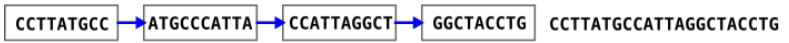
- Higher coverage, longer overlaps



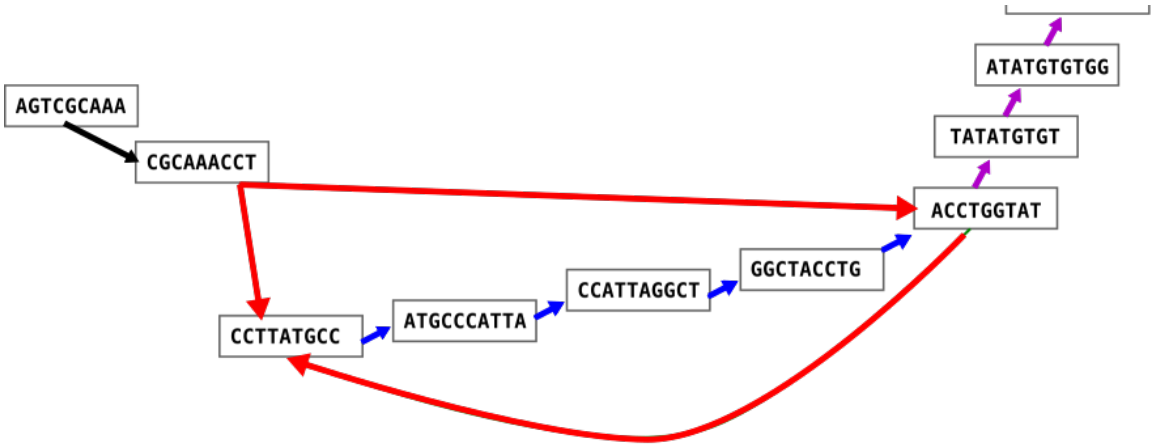
• Overlap graph



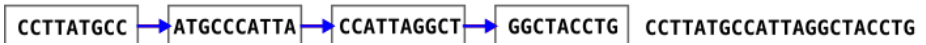
spurious assembly result



• Safe paths in an overlap graph



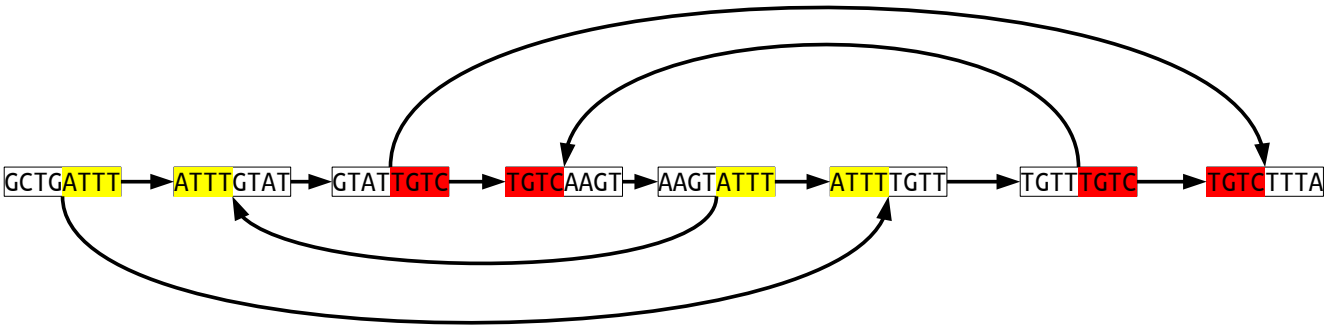
assembly result



• Multiple repeats

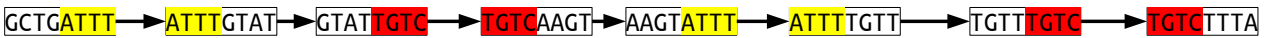
```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA
```

Overlap graph:



First solution

```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA Overlap graph:
```

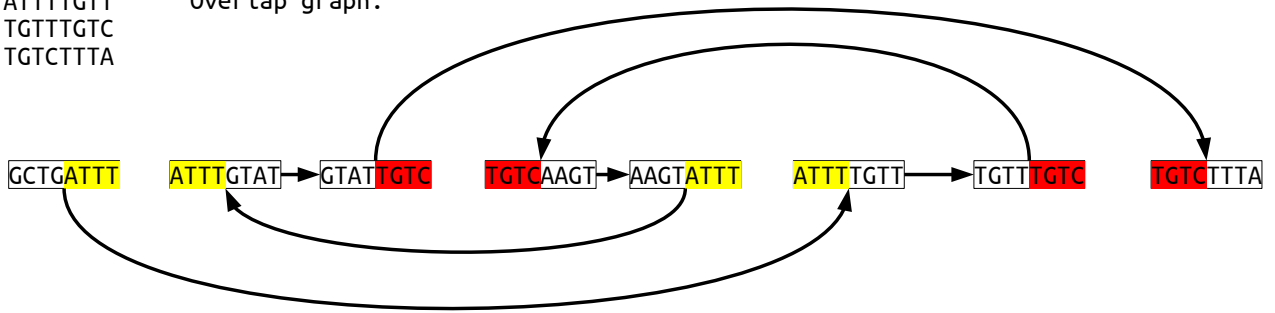



```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
```

• Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

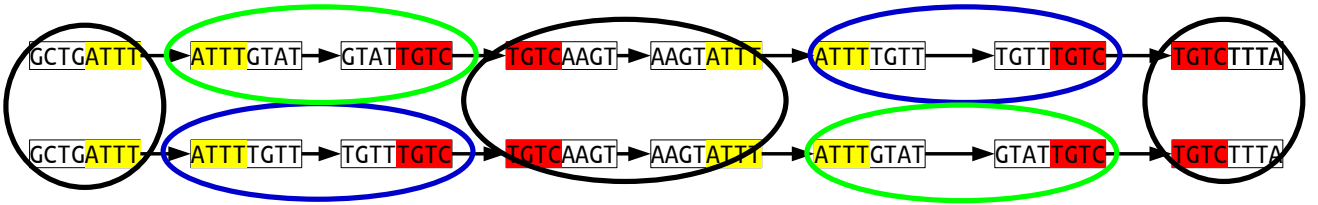


Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA

Those two solutions are indistinguishable

• Parsimonious solution: do not assemble

Possible assemblies:



Genome pieces:

GCTGATTT ATTTGTATTGTC TGTCAAGTATTT ATTTTGTTTGTC

Repeats lead to the fragmentation of the assembly

Genomes pieces that make **con*sensus across the differents solution are called** Con*tigs

The burden of assembly: genomic repeats

Amount of repeats larger than a given size in *E. coli* genome

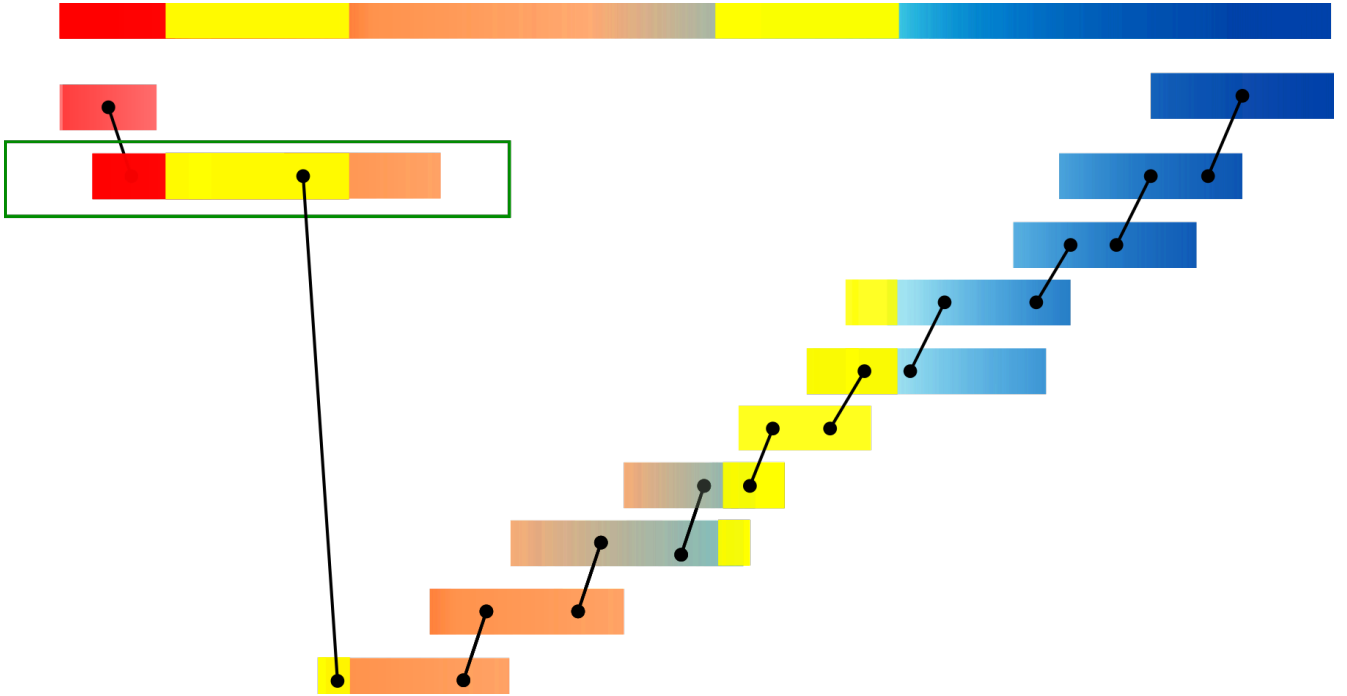
body

Genomic repeats are NOT random events

• Spanning reads solve repeat

Reads longer than the repeat “solve” it

• Spanning reads solve repeat (ii)



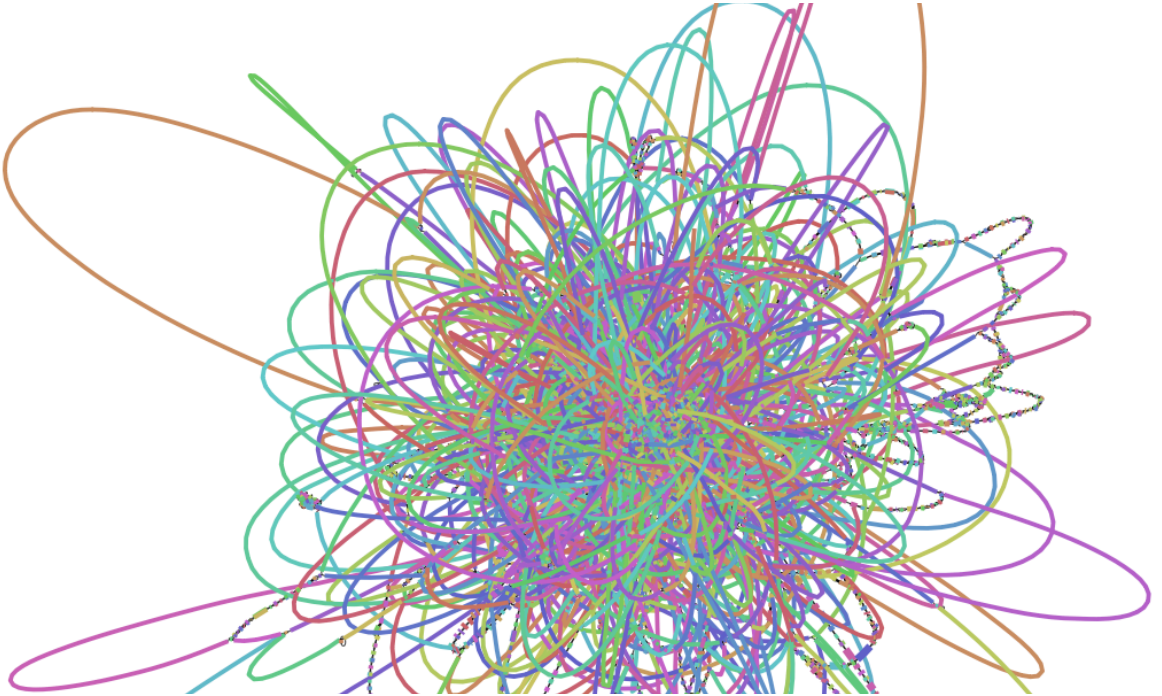
Spanning reads solve repeat (iii)

The graph becomes trivial to go traverse

• Read length matters

Read size=21

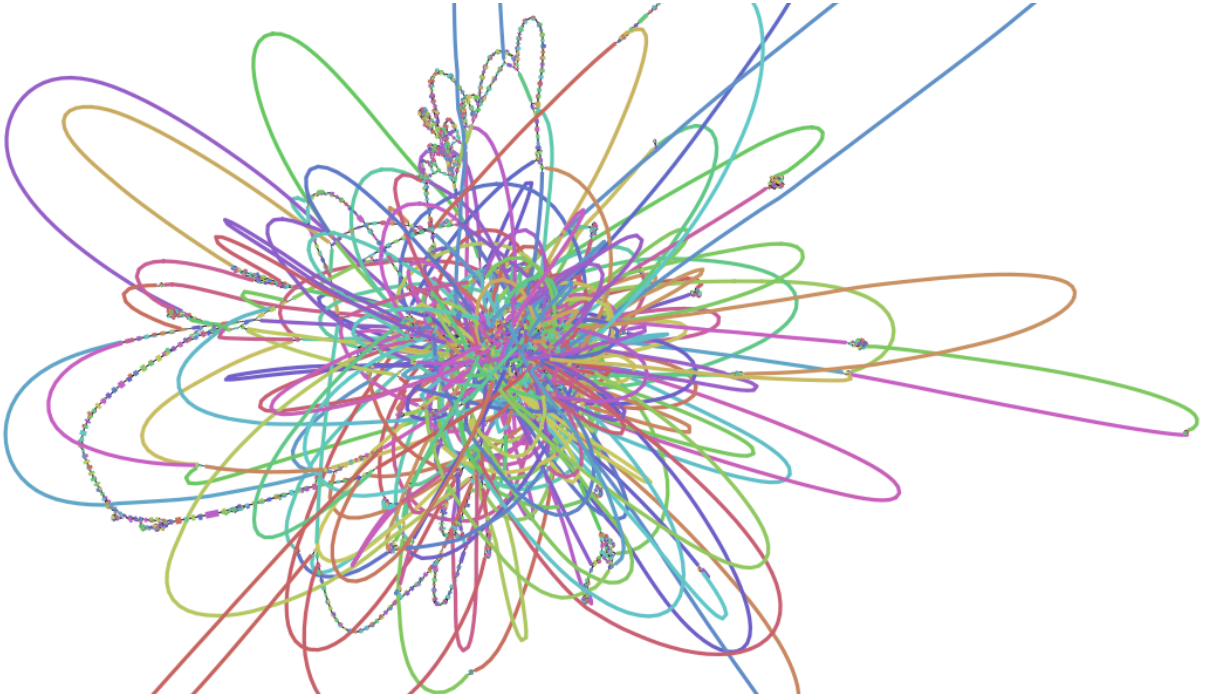
• Read length matters (ii)



• Read length matters

Read size=31

• Read length matters (ii)



• Read length matters

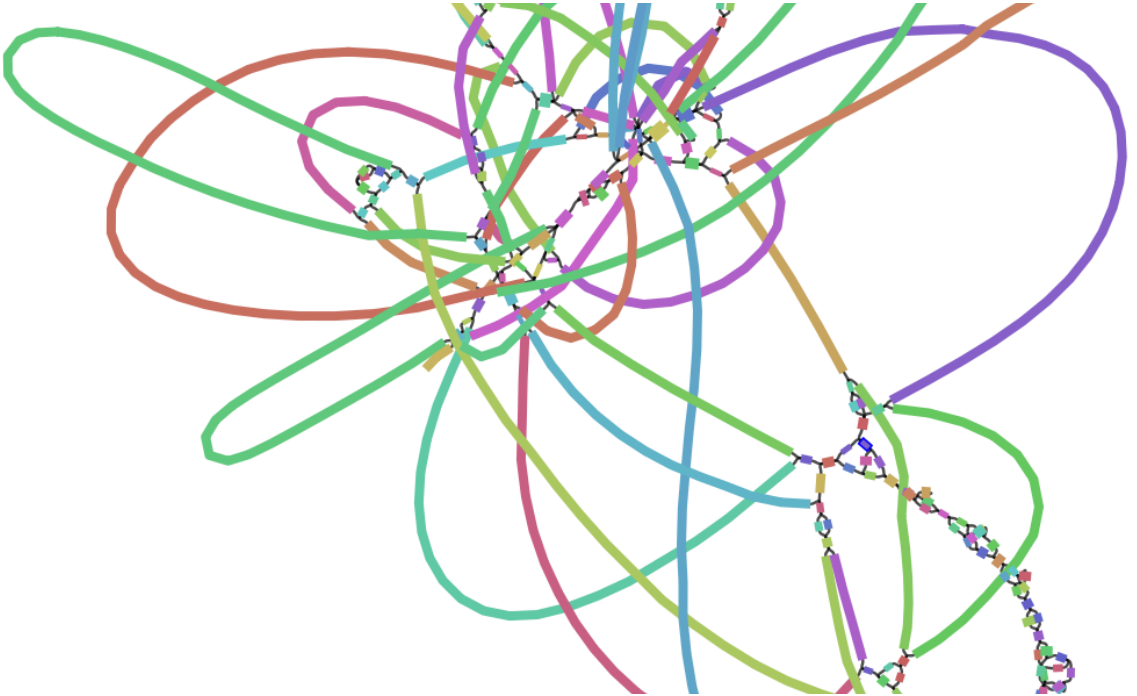
Read size=63

• Read length matters (ii)

• Read length matters

Read size=255

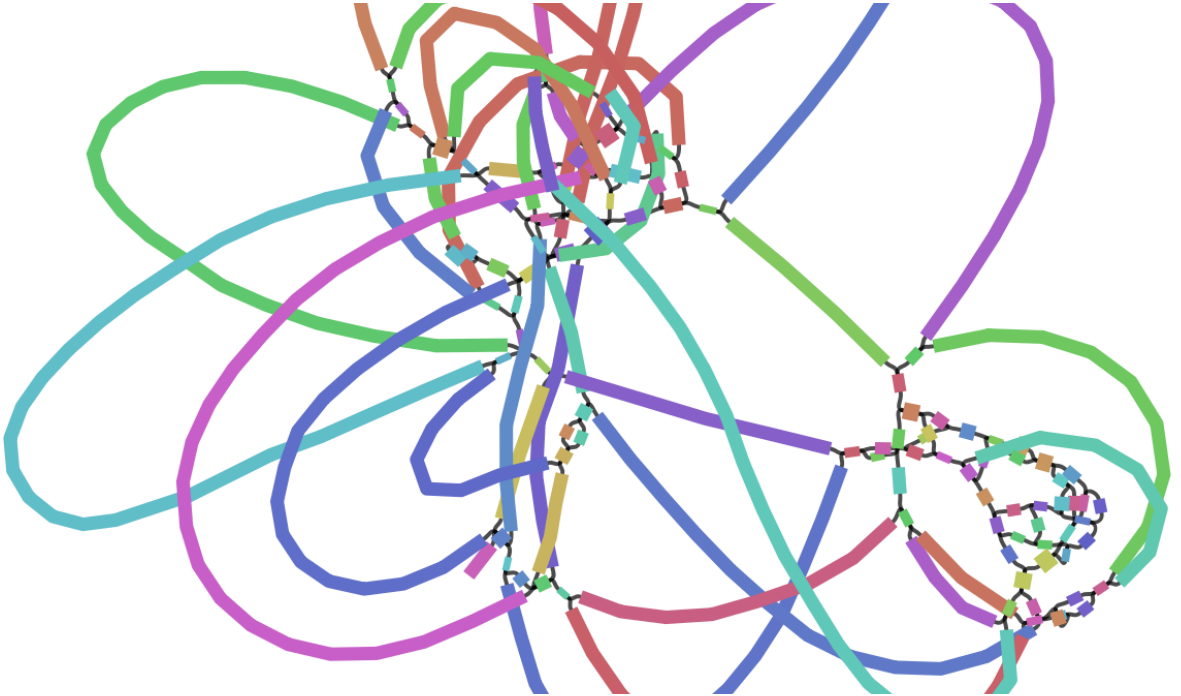
• Read length matters (ii)



• Read length matters

Read size=500

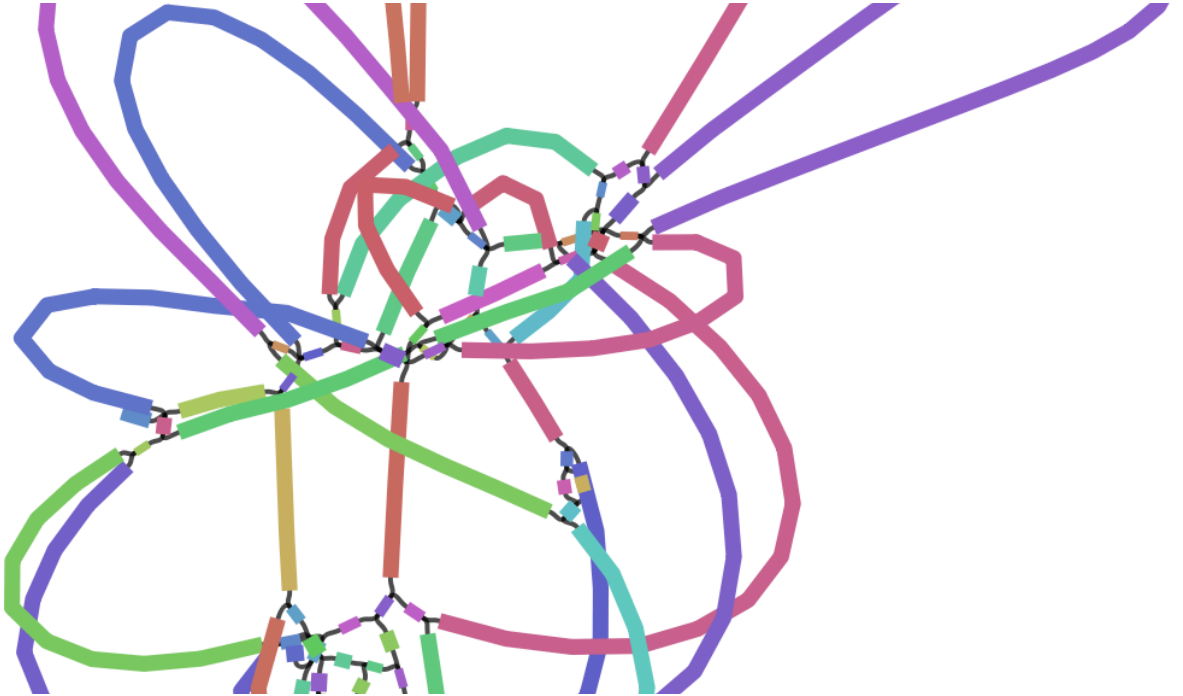
• Read length matters (ii)



• Read length matters

Read size=1000

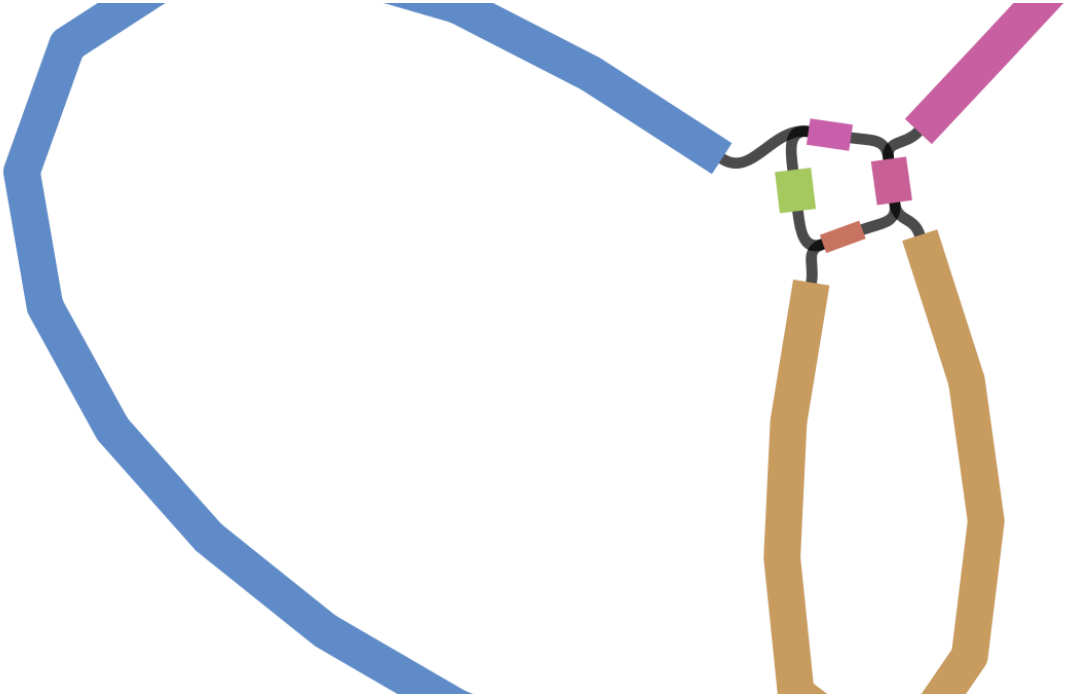
• Read length matters (ii)



• Read length matters

Read size=2000

• Read length matters (ii)



• Overlap graph burden: number of overlaps

For each base of the genome:

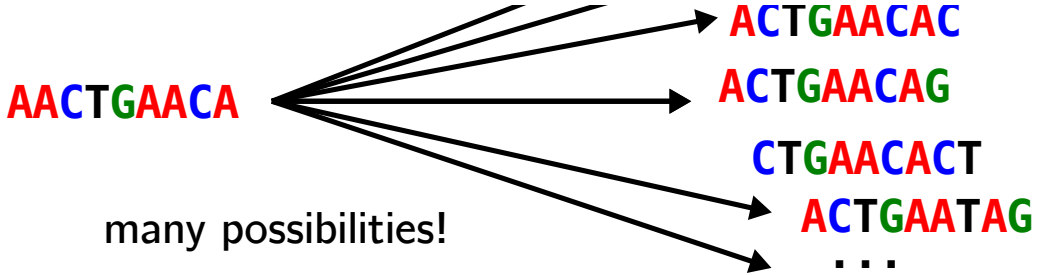
Read Coverage	Overlaps coverage
10	100
20	400
50	2,500
100	10,000

The amount of overlaps is not linear

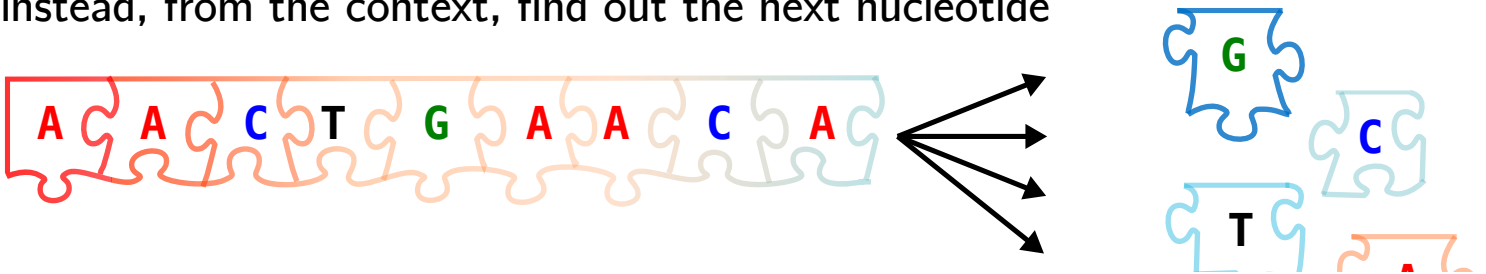
Linear: 2X data 2X time

Quadratic: 2X data 4X time

- **Another idea for assembly**



instead, from the context, find out the next nucleotide



De Bruijn graph

Kmer=node

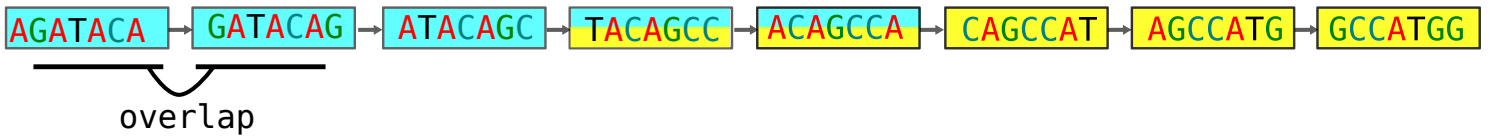
AGATACA + G + C + C + A

• de Bruijn graph assembly

Overlapping reads

AGATACAGCCA
TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATACAGCCATGG

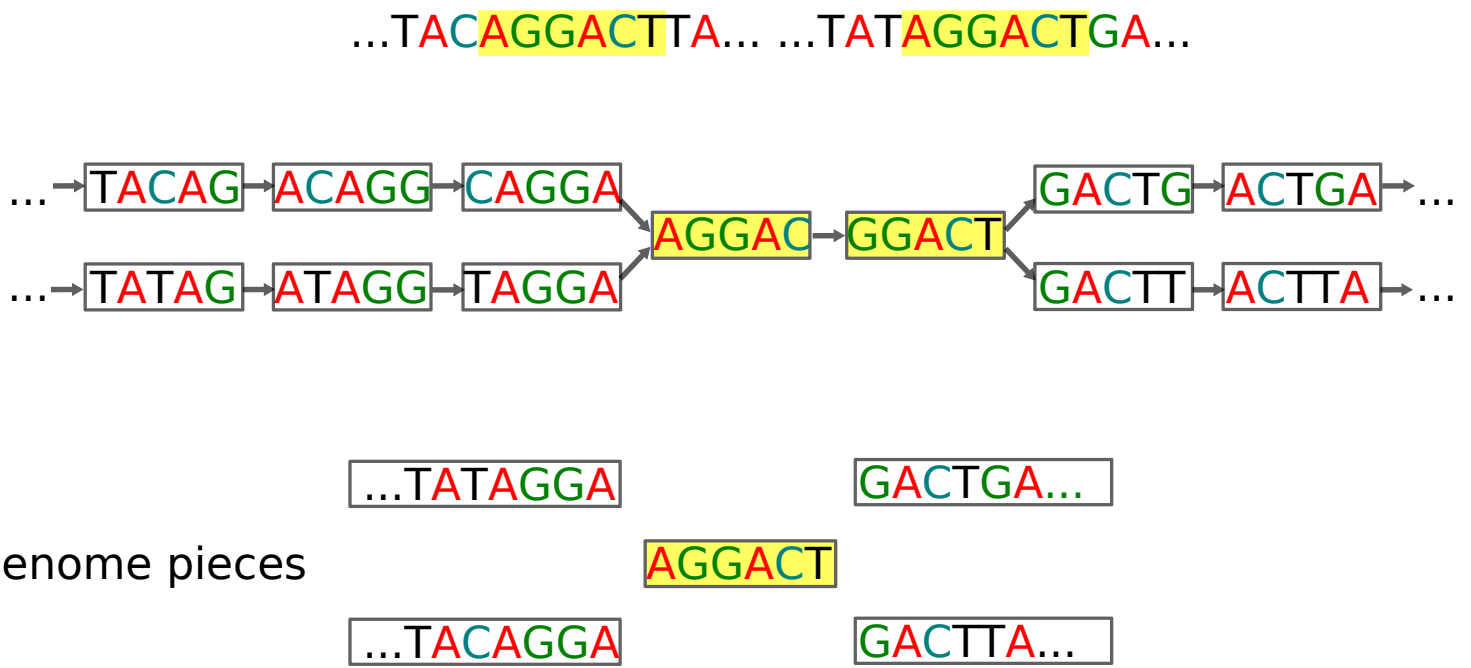
• de Bruijn graphs abstract redundancy

```
GATACAGCCAT
 ATACAGCCATG
  TACAGCCATGG
   ACAGCCATGGG
   ACAGCCATGGG
    CAGCCATGGGT
     AGCCATGGGTT
      GCCATGGGTTT
      GCCATGGGTTT
       CCATGGGTTTA
        CATGGGTTTAA
```

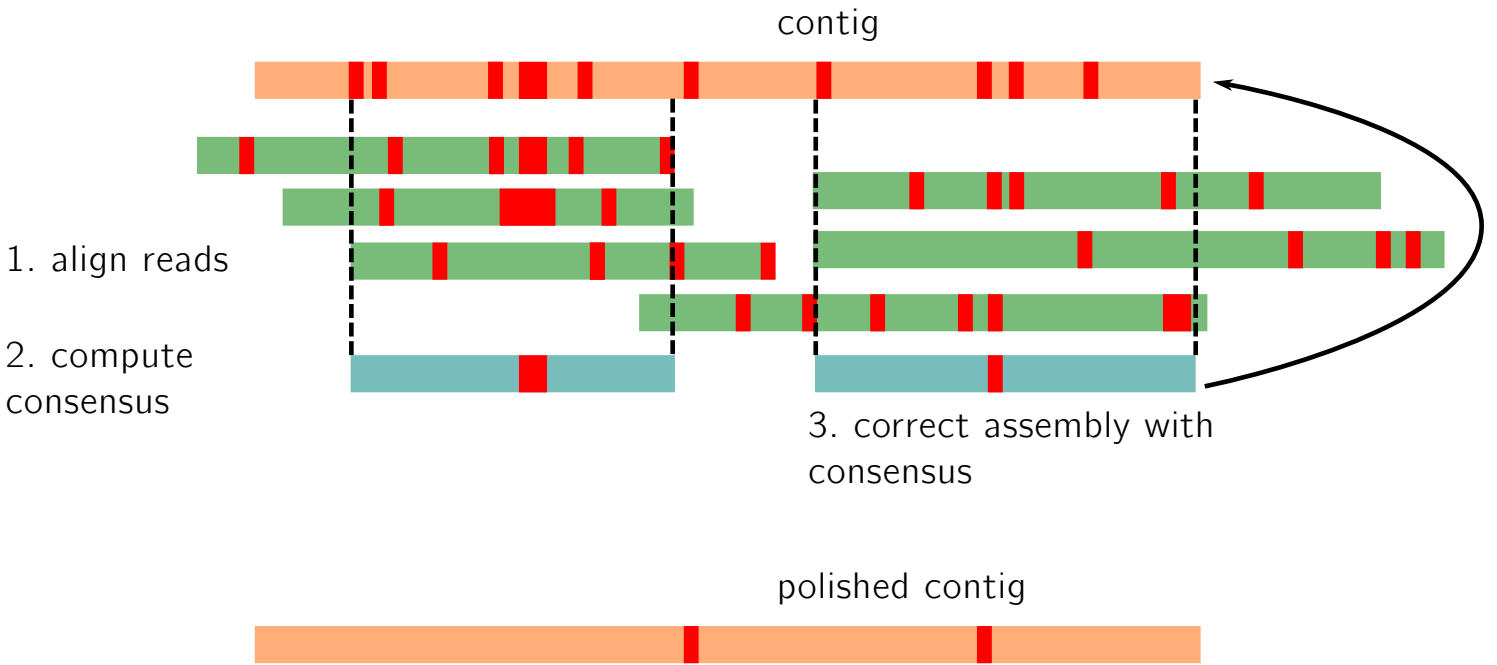
65 non distinct 7-mers in reads

14 **distinct** 7-mers in the de Bruijn graph

- **de Bruijn graph limitation**



• How to deal with sequenceing errors: polishing



• Handling errors with de Bruijn graphs

TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:

```
TAAGAAAGCTCTGAATCA
 AAGAAAGCTCTAAATCAAC
  AGAAAGCTCTGAATCAACG
   GAAAGCTCTGAATCAACGGA
    AAAGCTCTGAATCAACGGAC
     AAGCTCTGAATCAACGGACT
      AGCTCTGAATCAACGGACTG
       GCTCTGAATCAACGGTCTGC
        CTCTGAATCAACGGACTGCG
         TCTGAATCAACGGACTGCGA
```

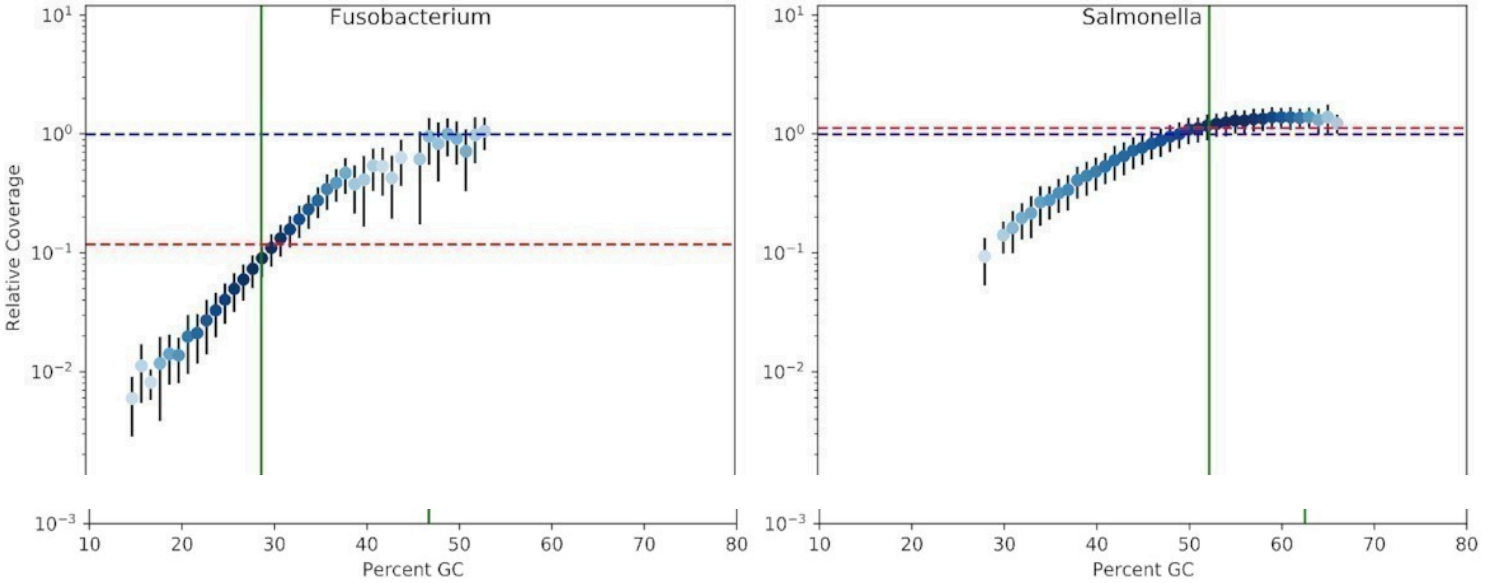
```
9 times   TCTGAAT
1 time    TCTAAAT

6 times           CAACGGA
1 time            CAACGGT
```

Handling errors with de Bruijn graphs (ii)

Erroneous k-mers are seen less than genomic ones

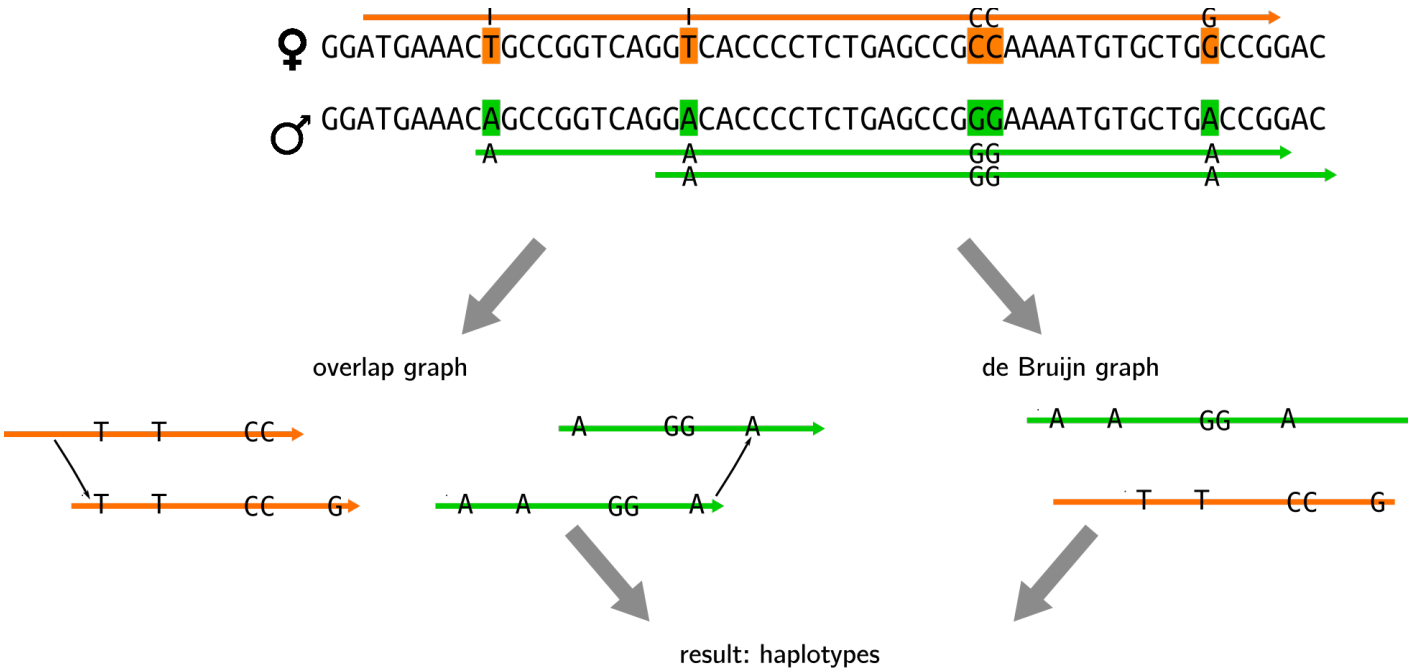
• K-mer spectrum

• GC bias


Fusobacterium
Salmonella
Relative Coverage
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10
20
30
40
50
60
70
80
Percent GC

GC bias (ii)

• Polyploid assembly

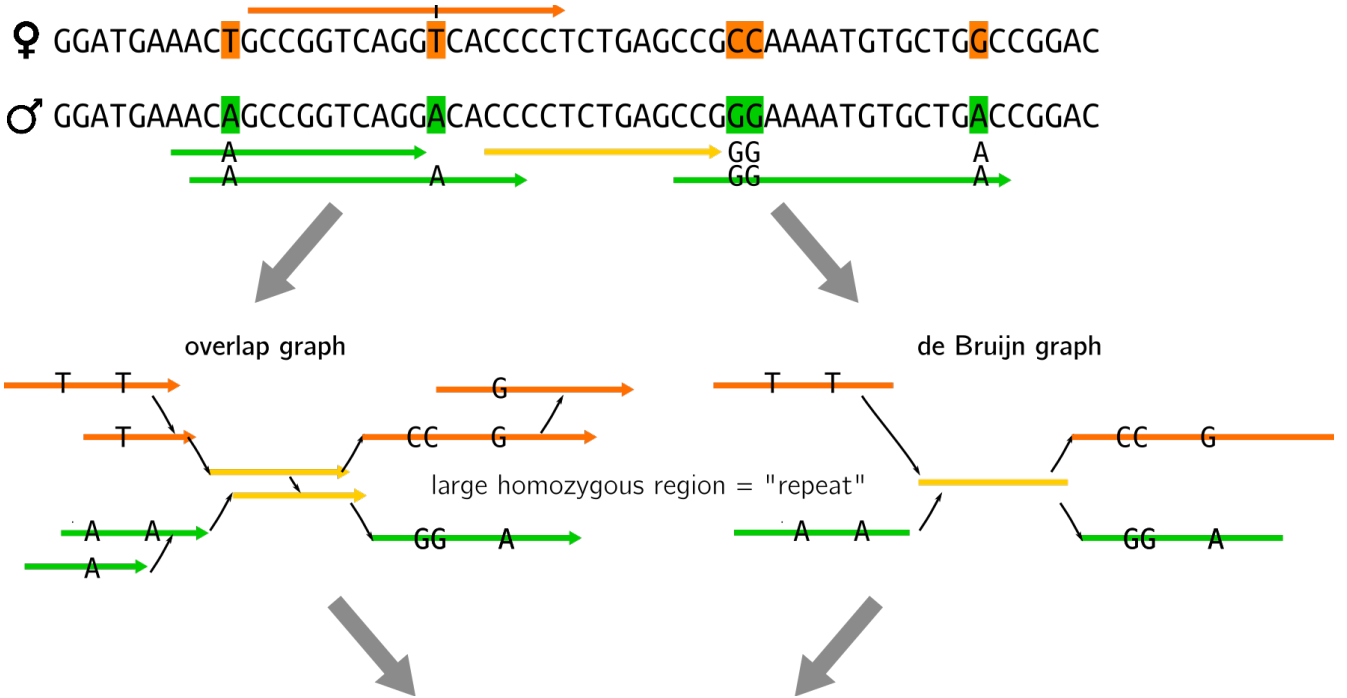


• Polyploid assembly (ii)

Haplotypes can be “phased”

Polyploid assembly (iii)

• Homozygous vs heterozygous regions

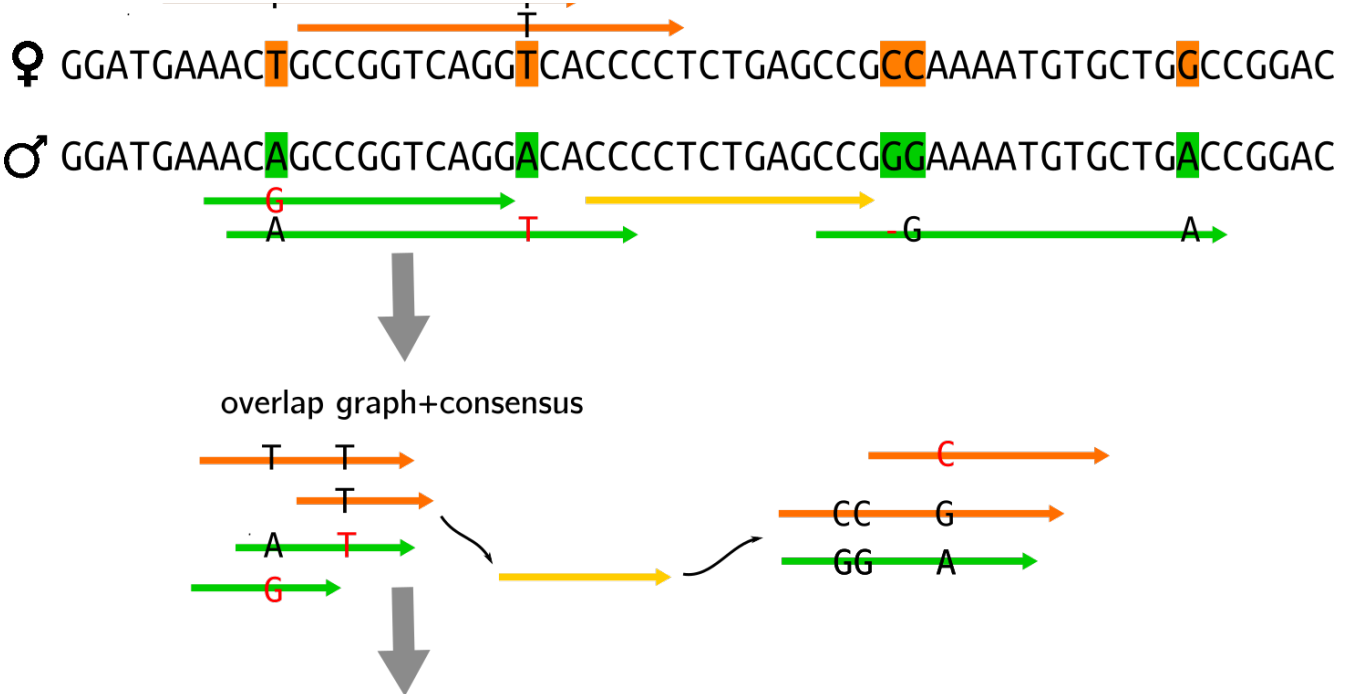


Homozygous vs heterozygous regions (ii)

Assembly concession: assembly can be fragmented due to homozygous region

• Homozygous vs heterozygous regions (iii)

• Sequencing error or heterozygosity?

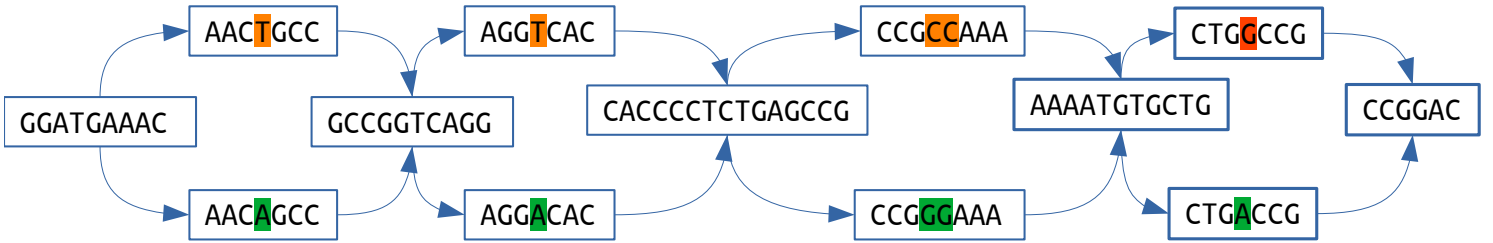


• Sequencing error or heterozygosity? (ii)

Assembly concession: “haploid” assembly due to errors

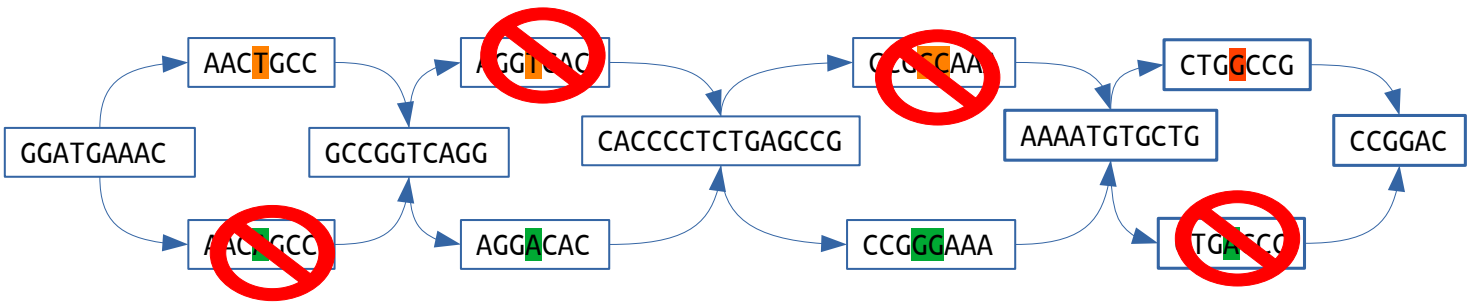
• Heterozygosity in de Bruijn graph

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGAAAATGTGCTGACCGGAC



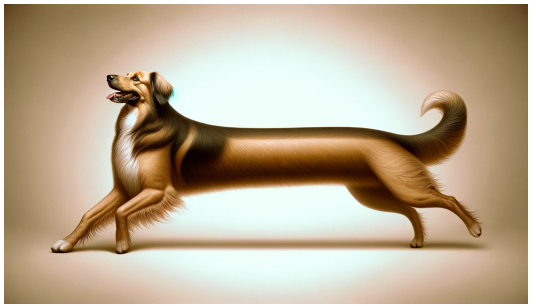
• Bubble crushing

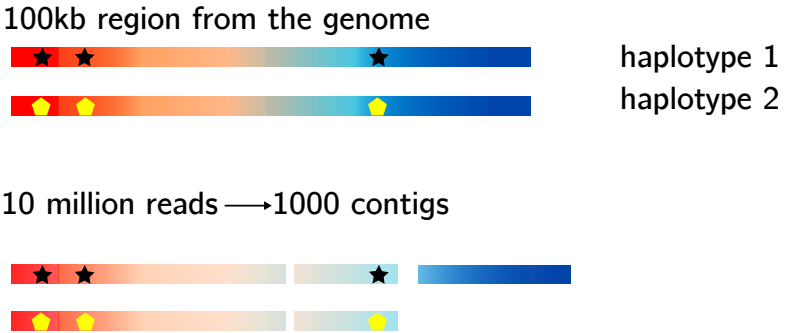
♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Assembly:

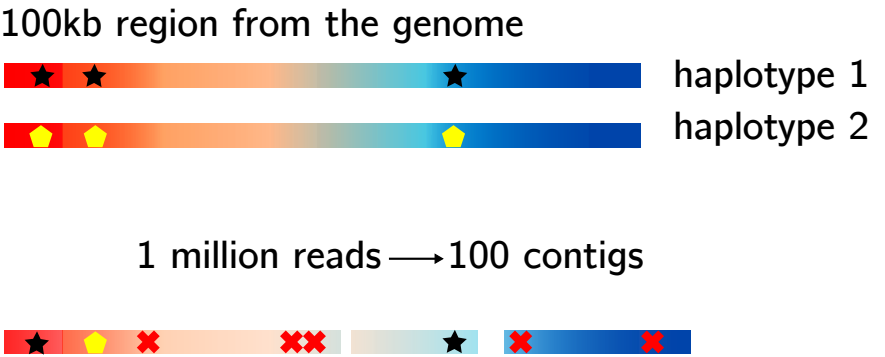
Long “perfect” reads: HiFi or accurate Nanopore





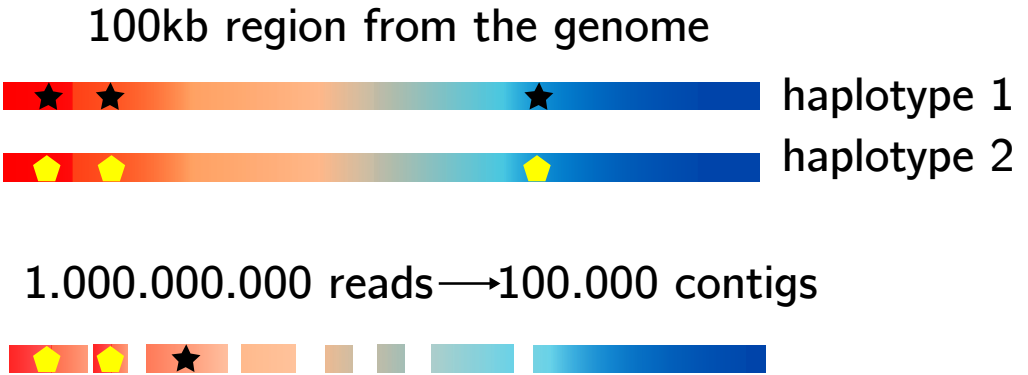
- Contigs can reach the chromosome’s order of magnitude in length (megabases)
- Breaks due to large repeats
- Haplotypes can be partially reconstructed
- Can be assembled with either de Bruijn graph or overlap graph (cheap to assemble)

Very long noisy reads: Oxford Nanopore and Pacific Bioscience



- Contigs can reach the chromosome's order of magnitude in length (megabases)
- Breaks due to very large repeats
- Contigs are chimeras of haplotypes
- Overlap graph (costly to assemble)

Accurate short reads: Illumina



- Very fragmented assembly of short contigs (mostly below 100kb)
- Very high base accuracy
- Contigs are chimeras of haplotypes
- Can miss extreme GC content
- De Bruin graph (cheap to assemble)

Expensive reads: Sanger

Expensive reads: Sanger (ii)

Billion $ project

Challenge for assembly: Scalability

- Human Genome project (2001)
- 1000 Genomes project (2015)
- 10k Genomes project (2016)
- 100k Genomes project (2018)
- 500K UK genomes (2023)

Challenge for assembly: Scalability (ii)

Many ambitious sequencing projects beyond human: Earth biogenome project, Vertebrate genome project ...

History

How long to assemble a human genome?

- Sanger: MANY CPU **years**
- Illumina (Overlap graph): 2 CPU **months**
- Illumina (De Bruijn graph): A CPU **day**
- Long reads (Alignment): 2 CPU **years**
- Long reads (Anchors chaining): 20 CPU **days**
- HiFi (Anchors chaining): 2 CPU **days**
- HiFi (De Bruijn graph): A CPU **hour**

Algorithms and data structures matter!

Also long and precise reads are easier to assemble

• Challenge for assembly: Telomere to telomere assembly

• Challenge for assembly: Telomere to telomere assembly (ii)

Telomere-to-Telomere consortium

Has produced in 2021 a complete human genome with one contig per chromosome !

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- Arima Genomics HiC
- BioNano DLS
- 100 authors from 50 labs

Telomere-to-Telomere diploid human reference

T2T-YAO released in 2023 a complete human genome with one contig per chromosome !

- 92x PacBio HiFi
- 336x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 584x Arima Genomics HiC
- BioNano DLS
- Illumina HiSeq 150bp for the son and parents (with 278x and 116x coverage, respectively).

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)

• The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)
- Axolotl (32Gb)

The human genome is not THAT hard

Hall of fame of largest assembled genomes of their time:

- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)

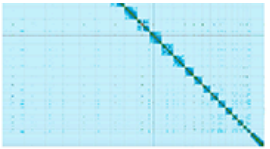
• The human genome is not THAT hard

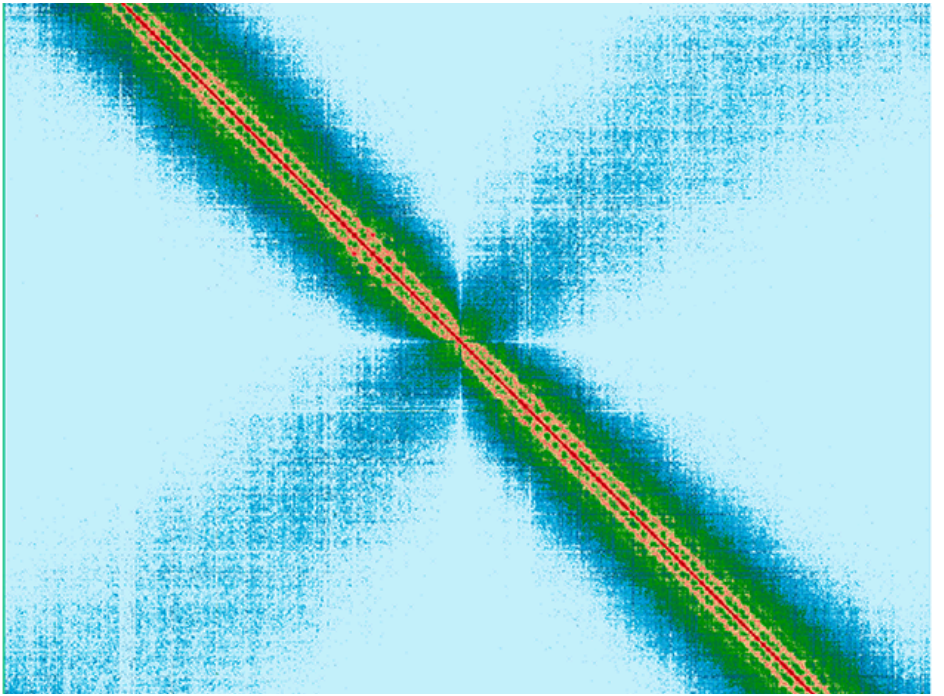
Hall of fame of largest assembled genomes of their time:

The human genome is not THAT hard (ii)

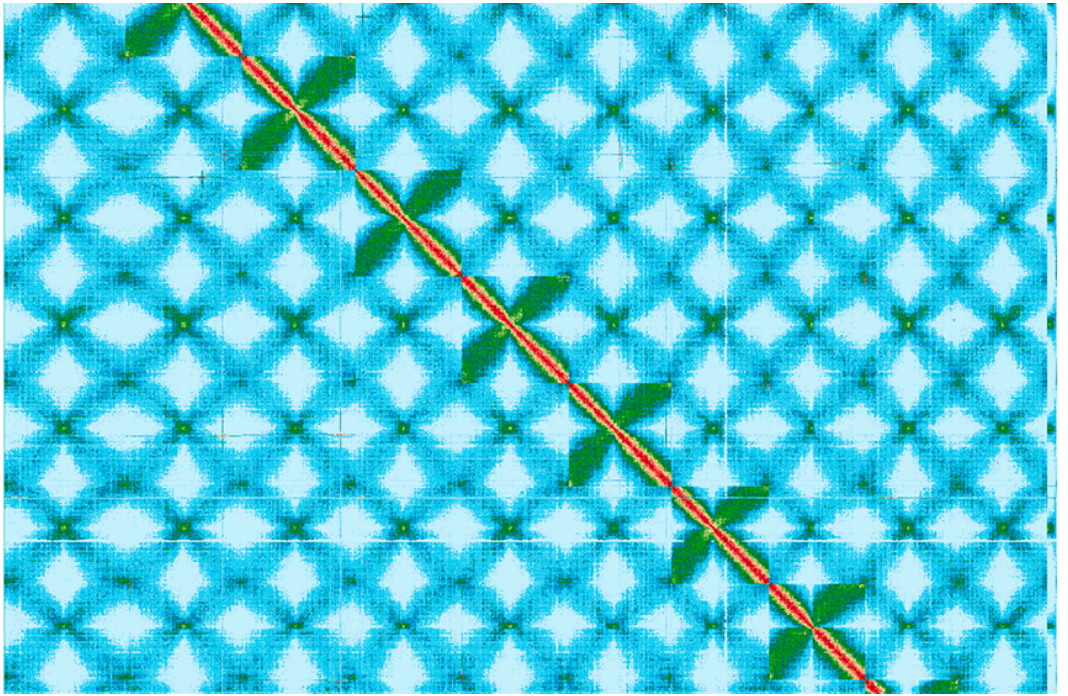
- Pine (20Gb)
- Axolotl (32Gb)
- Lungfish (43Gb)
- Mistletoe (90Gb)
- Metagenomes ...

• The human genome seems small





• The human genome seems really small



Challenge 3: Base level accuracy

High precision is required to distinguish highly related sequence such as

- Divergent repeats(intra chromosomes)
- Heterozygosity (inter chromosomes)

Sequencing technologies are improving

body

What is left to do

Megabase level assembly

body

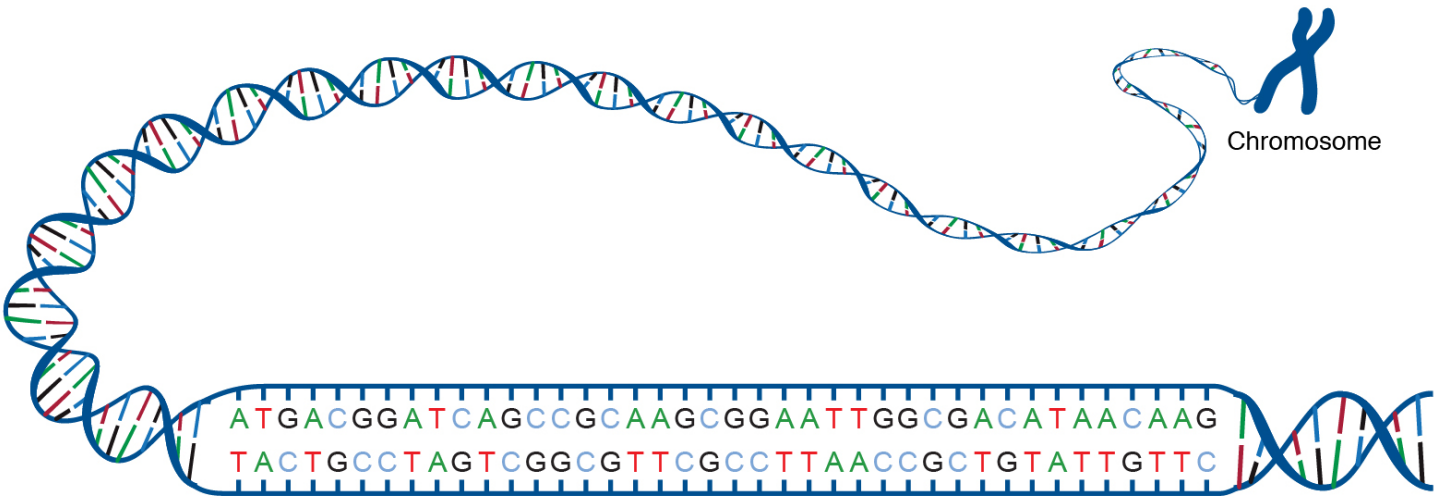
T2T assembly

body

Diploid assembly

body

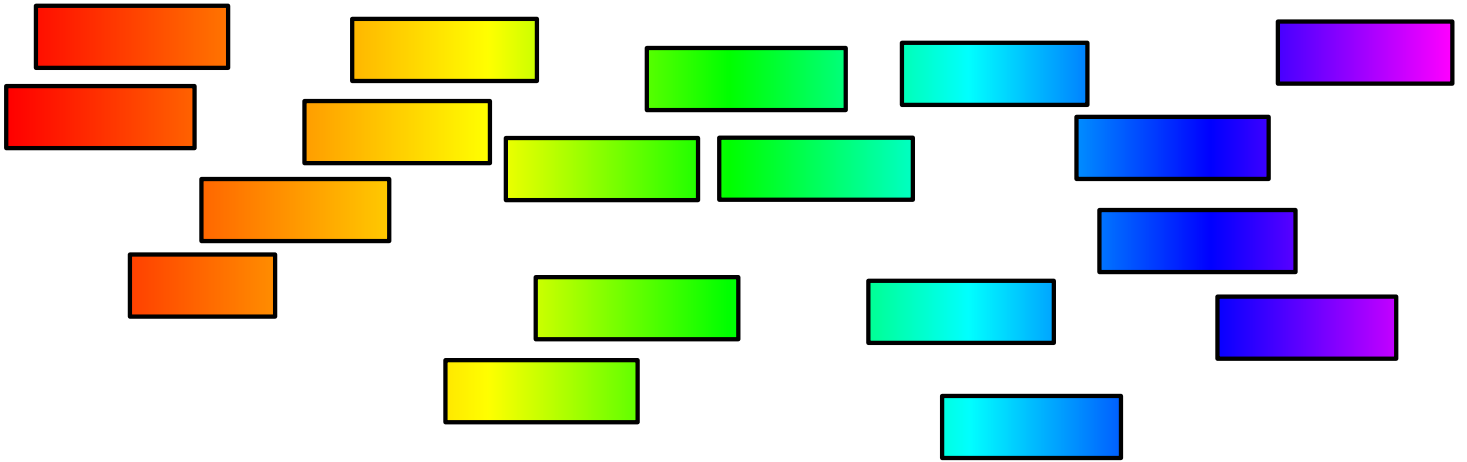
• Accessing a genome



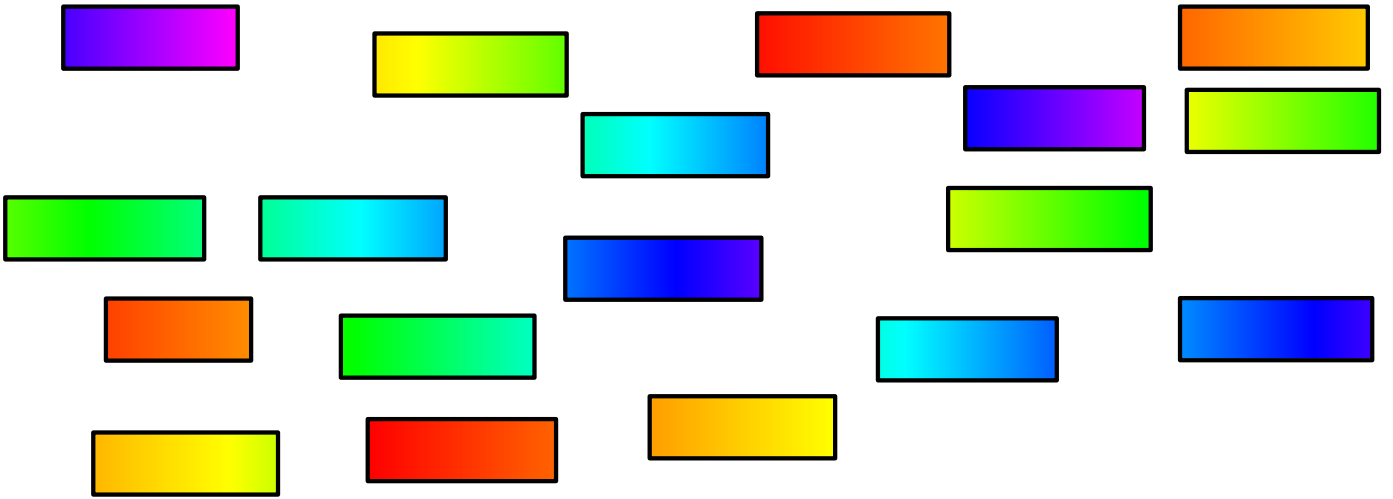
• Accessing a genome (ii)

From https://www.genome.gov/genetics-glossary/acgt

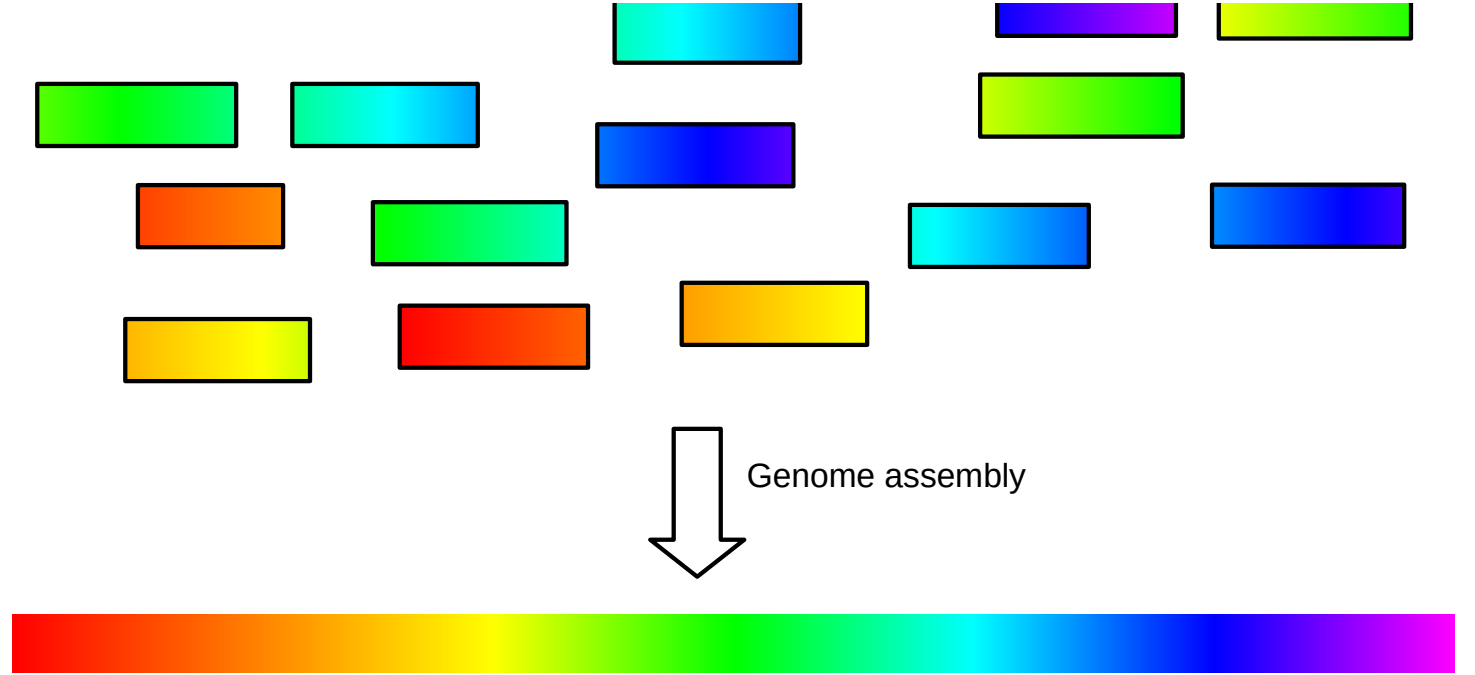
- **Reads are subsequences from the genome**



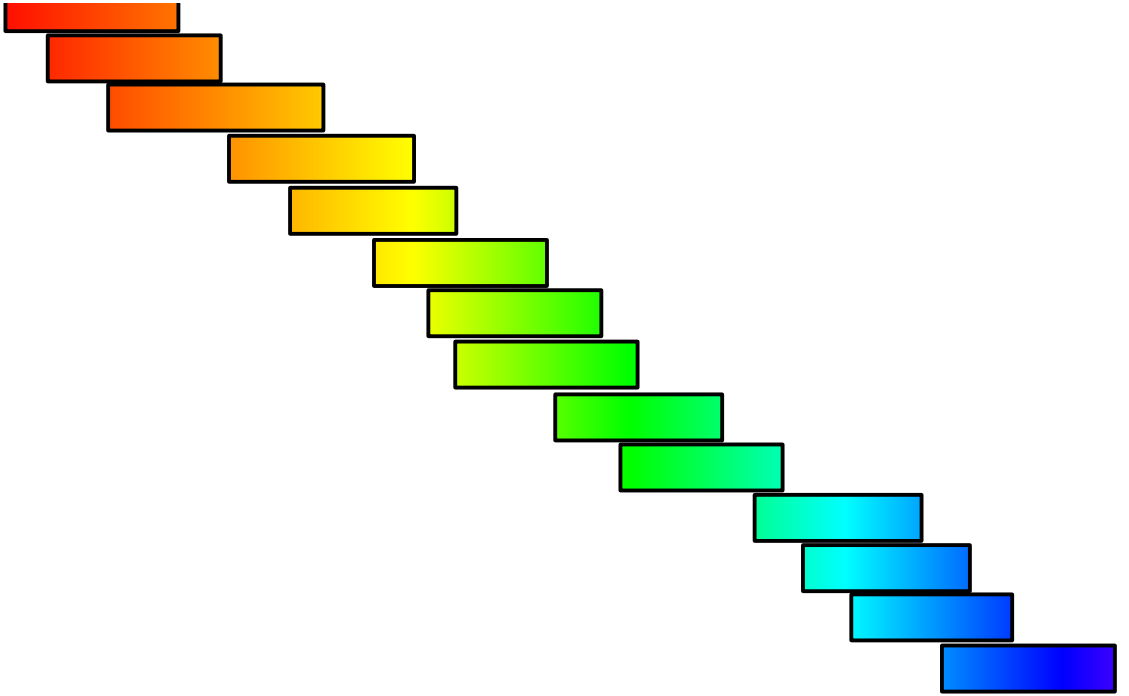
- **Reads are shuffled subsequences from the genome**



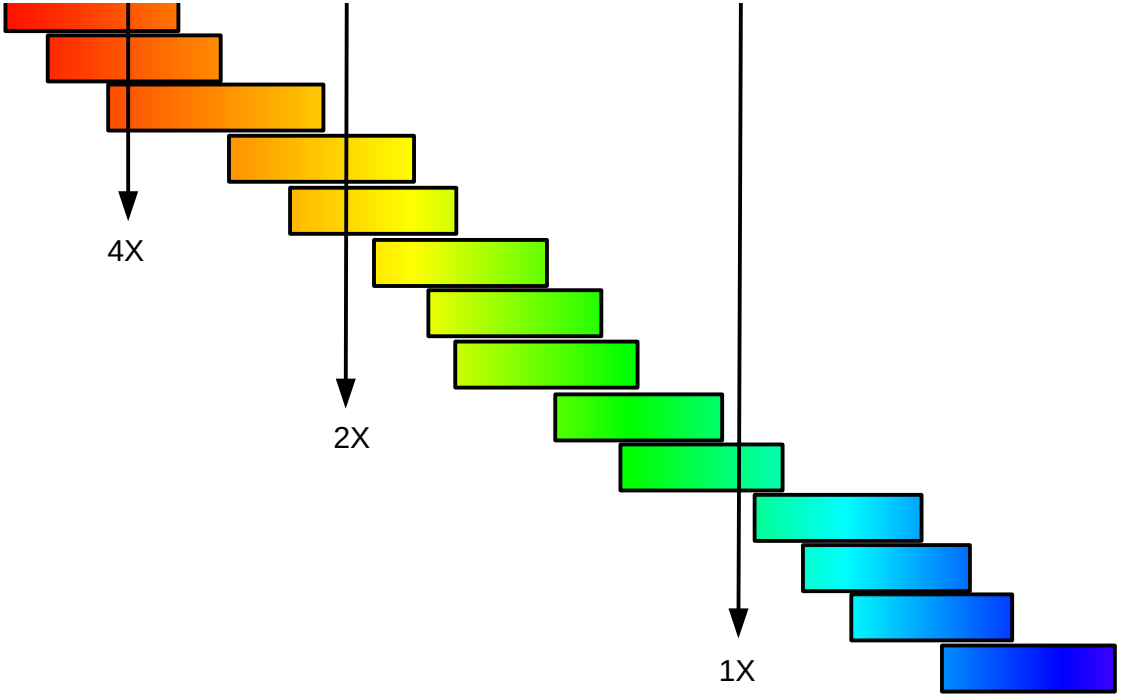
• Genome assembly task



• Using read overlaps

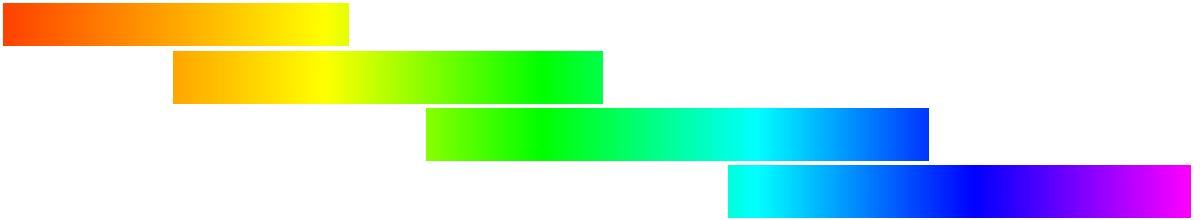


Genome sequencing: coverage



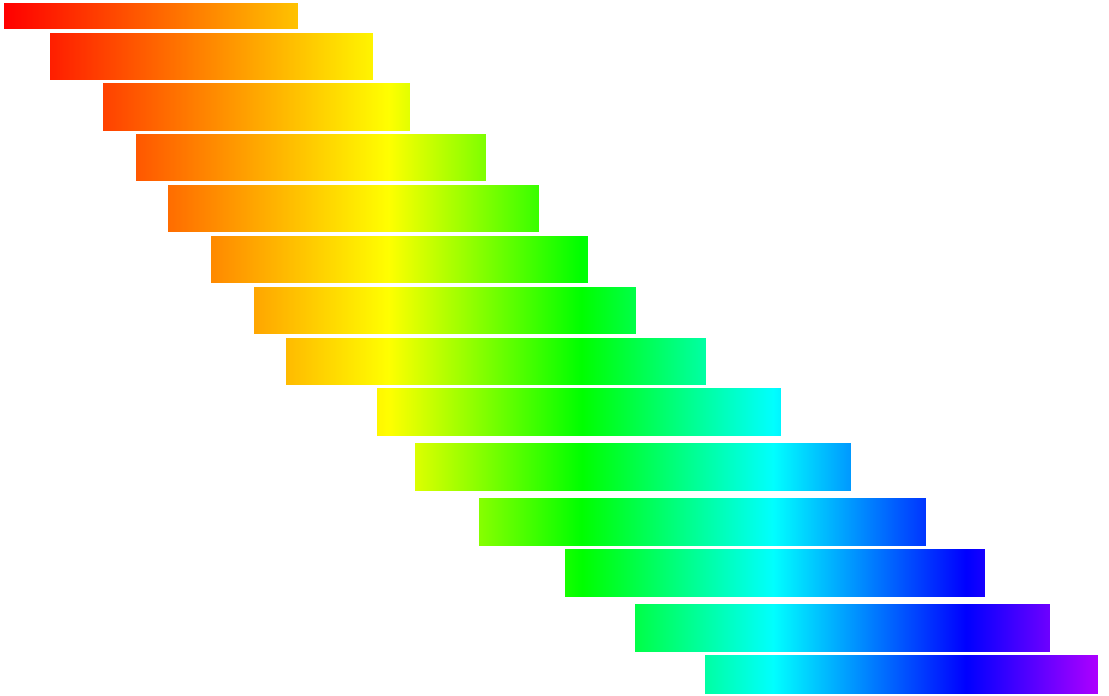
Genome sequencing: coverage

Genome sequencing: coverage

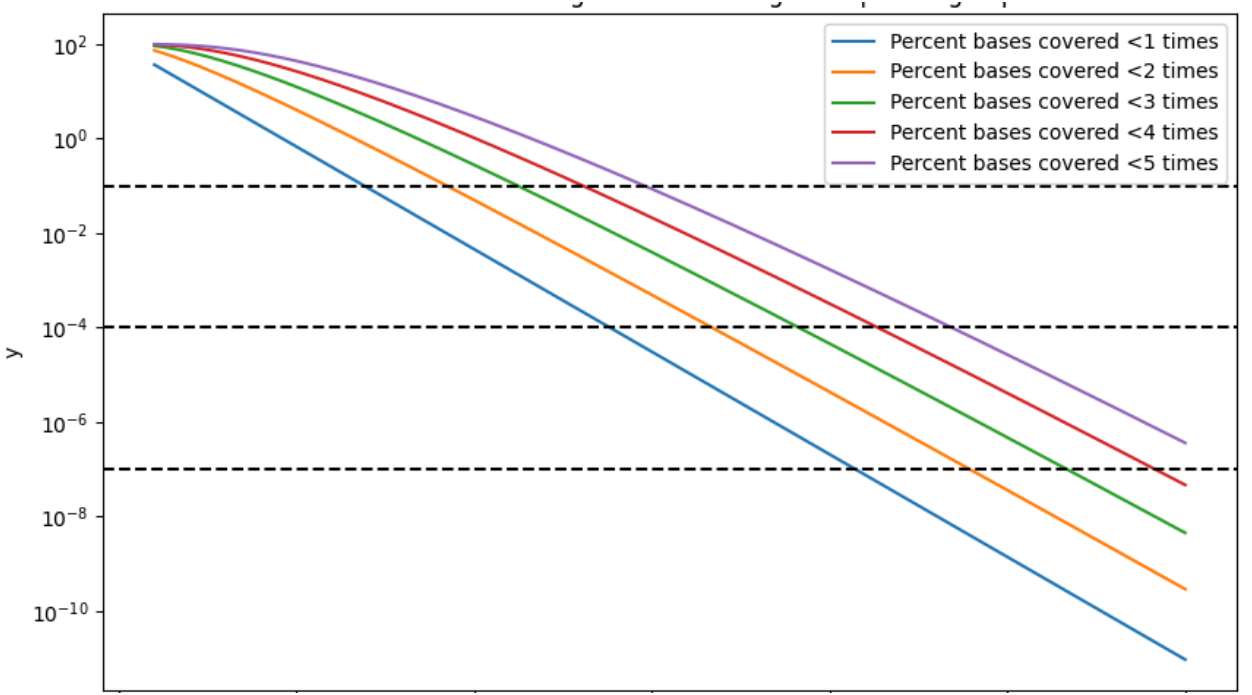


Genome sequencing: coverage

Genome sequencing: coverage



• Genome sequencing: coverage



Genome sequencing: coverage (ii)

30-100X are often required for assembly projects

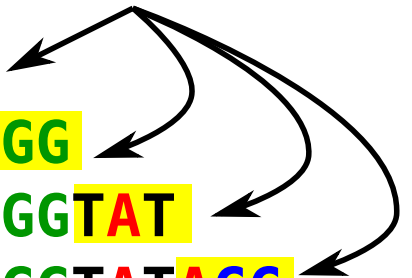
- **Which overlap?**

```
5:  ACGGTATACC


1:  ATCGGTATCG                    Overlap length:7
2:     GGTATCGTTA

1:  ATCGGTATCG                    Overlap length:4
3:        ATCGTTACGG

1:  ATCGGTATCG                    Overlap length:1
4:            GTTACGGTAT

1:  ATCGGTATCG                    No overlap
```

- Assembly idea number 1: assemble the longest overlaps

```
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATACC

Best overlaps:
1: ATCGGTATCG
2:    GGTATCGTTA
3:       ATCGTTACGG
4:          GTTACGGTAT
5:             ACGGTATACC
```

supplementary information brought by each read



- **Your time to shine!**

Your read set:

1: ATTTACGGGT
2: TTACGGGTGG
3: ACGGGTCCTT
4: GTCCTTTCTT
5: TTTCTTACGG

For each read:
    Find the best overlap (length>5)
    Merge the two reads

- The Greedy solution

```
ACGGGTCCTT
    GTCCTTTCTT
        TTTCTTACGG
```

```
Output “genome”
ATTTACGGGTGG
ACGGGTCCTTTCTTACGG
```

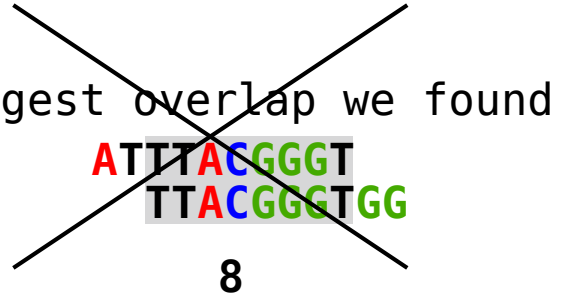
• The actual solution

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:

```
ATTTACGGGT
    ACGGGTCCTT
    6   GTCCTTTCTT
            TTTCTTACGG
                TTACGGGTGG
```



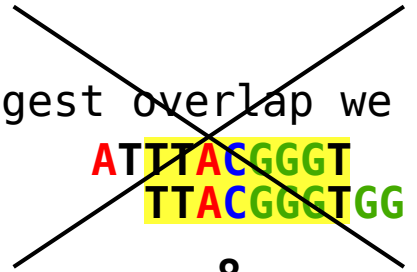
- **What happened?**

The actual genome:

ATTTACGGGTCCTTTCTTACGGGTGG

How the reads should be ordered:

```
ATTTACGGGT
   ACGGGTCCTT
   6   GTCCTTTCTT
          TTTCTTACGG
             TTACGGGTGG
```



ATTTACGGGTGG
not in the genome

ACGGGTCCTTTCTTACGG
not in the genome

• Do we expect many repeats?

Probability to have NO repeated word of size 31 in a 5 megabases genome

body

From https://en.wikipedia.org/wiki/Birthday_problem

The burden of assembly: genomic repeats

Amount of repeats larger than a given size in *E. coli* genome

body

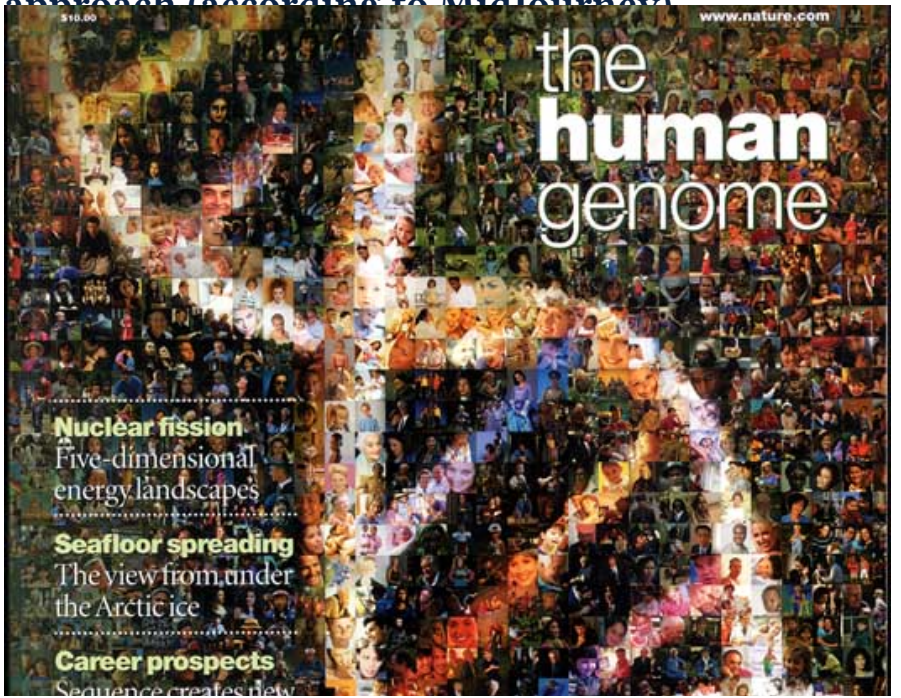
Genomic repeats are NOT random events

Greedy assemblers

- Simple and efficient scheme
- Rely on **local** best choice (greedy)
- May create errors because of local choices when there are repeats

• Greedy assemblers (ii)

- **History: the human genome project while finding a successor to the greedy approach (according to MidJourney)**



• Graph representation

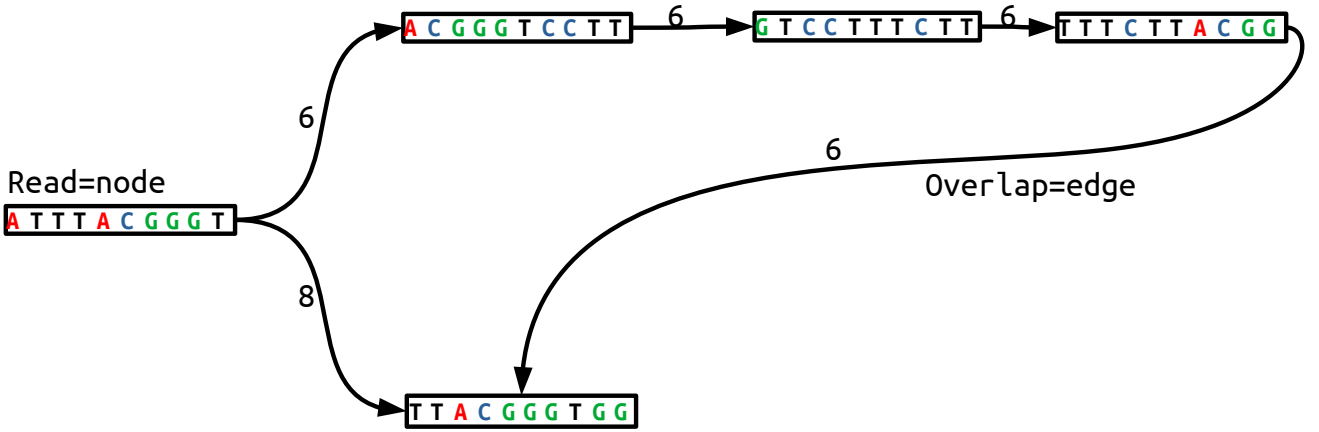
a node is a sequence

an arc oriented between
a source node and
a sink node

an arc means there is an overlap between the end of the source node and the beginning of the sink node

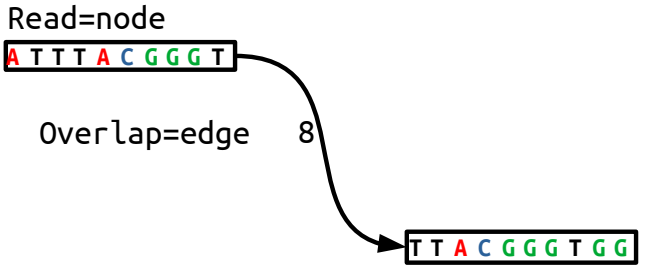
• Assembly idea number 2: consider all overlaps

Overlap graph:



• Greedy solution

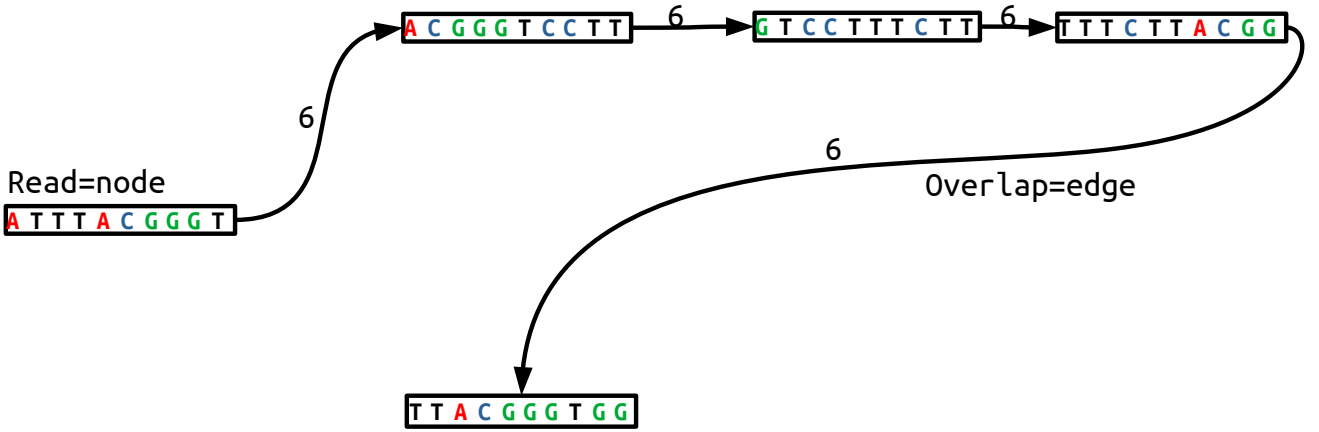
Overlap graph:



Greedy assembly output:

• One piece solution

Overlap graph:

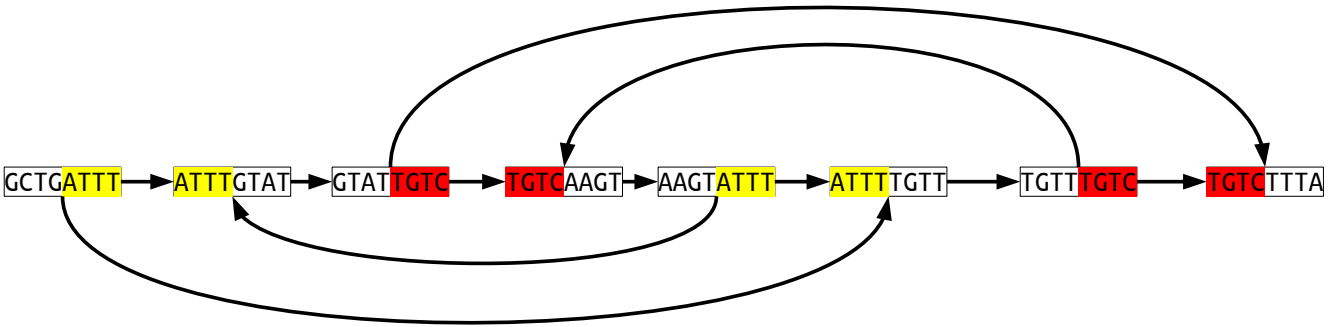


Overlap graph output:

• Multiple repeats

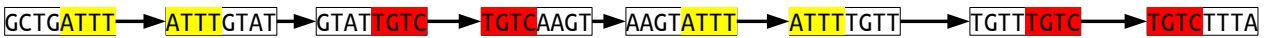
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:



First solution

```
Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA Overlap graph:
```

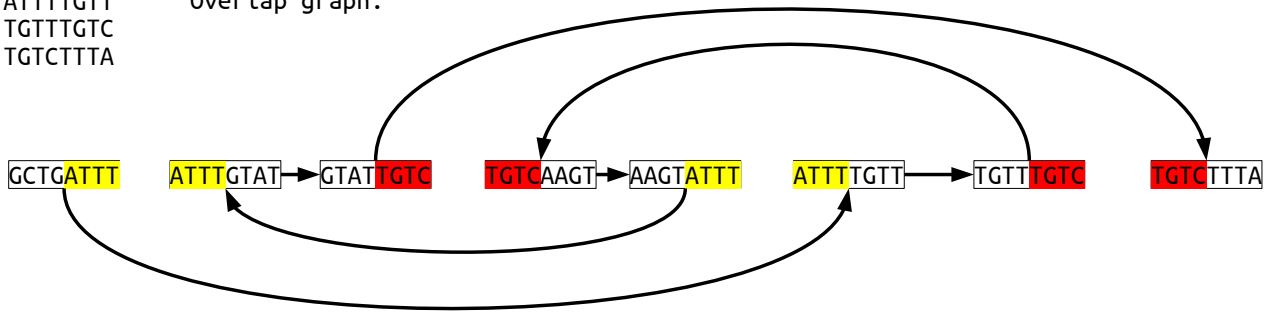



```
Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
```

• Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

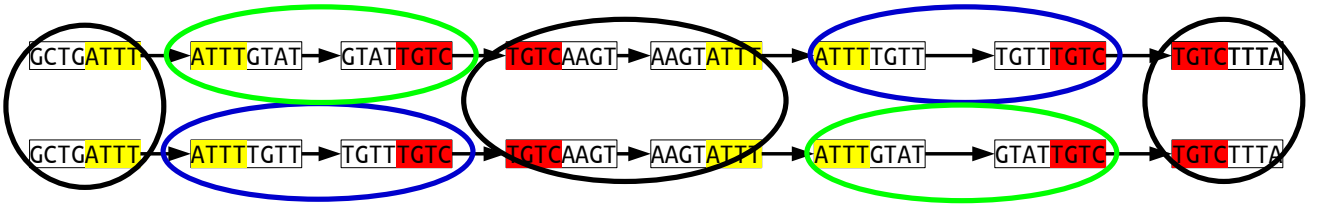


Possible assemblies:
GCTGATTTGTATTGTCAAGTATTTTGTTTGTCTTTA
GCTGATTTTGTTTGTCAAGTATTTGTATTGTCTTTA

Those two solutions are indistinguishable

• Parsimonious solution: do not assemble

Possible assemblies:

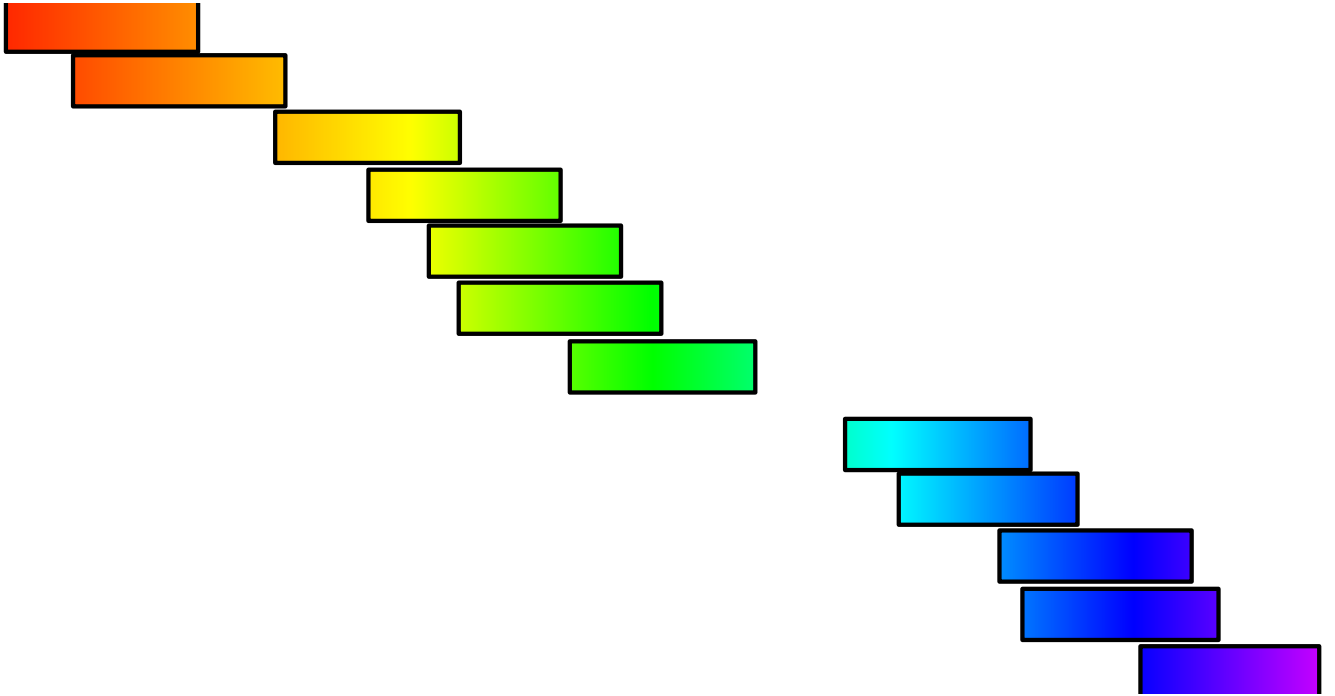


Genome pieces:

GCTGATTT ATTTGTATTGTC TGTCAAGTATTT ATTTTGTTTGTC

Repeats lead to the fragmentation of the assembly

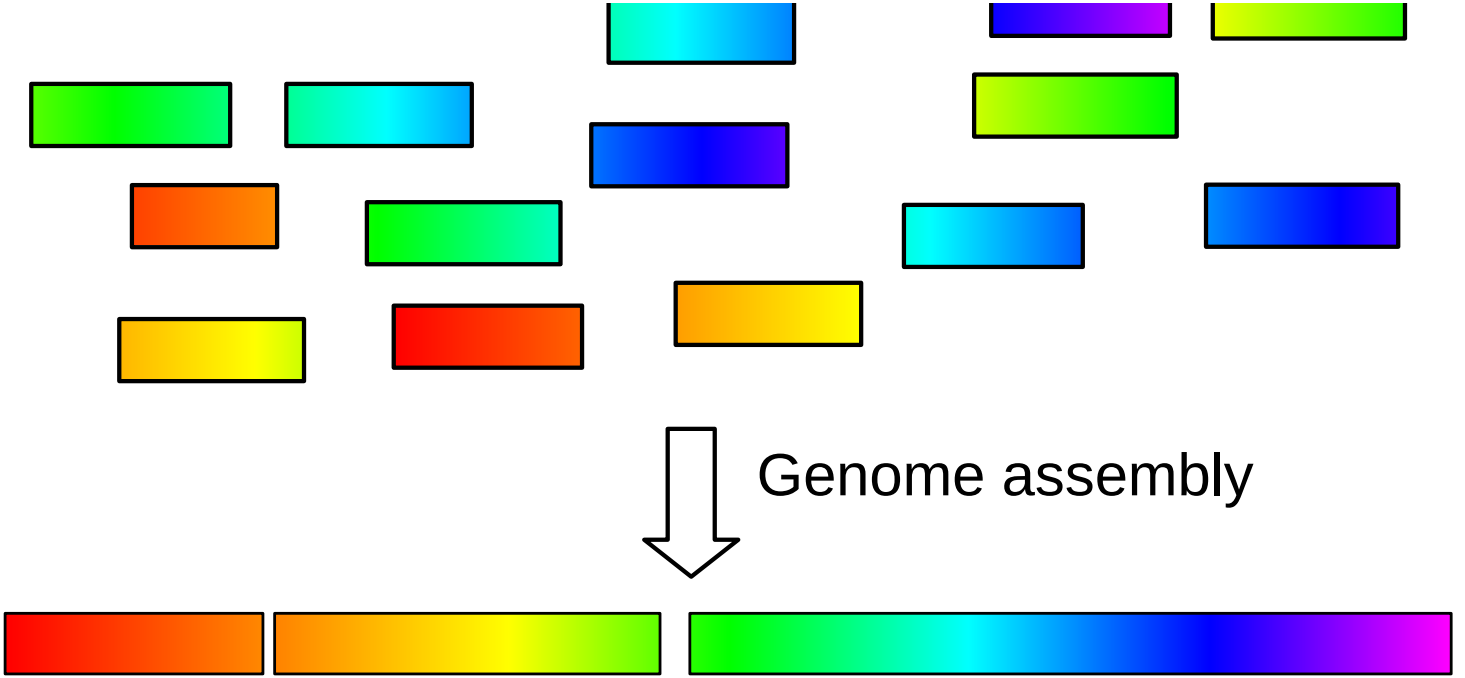
- **Missing information also fragments the assembly**



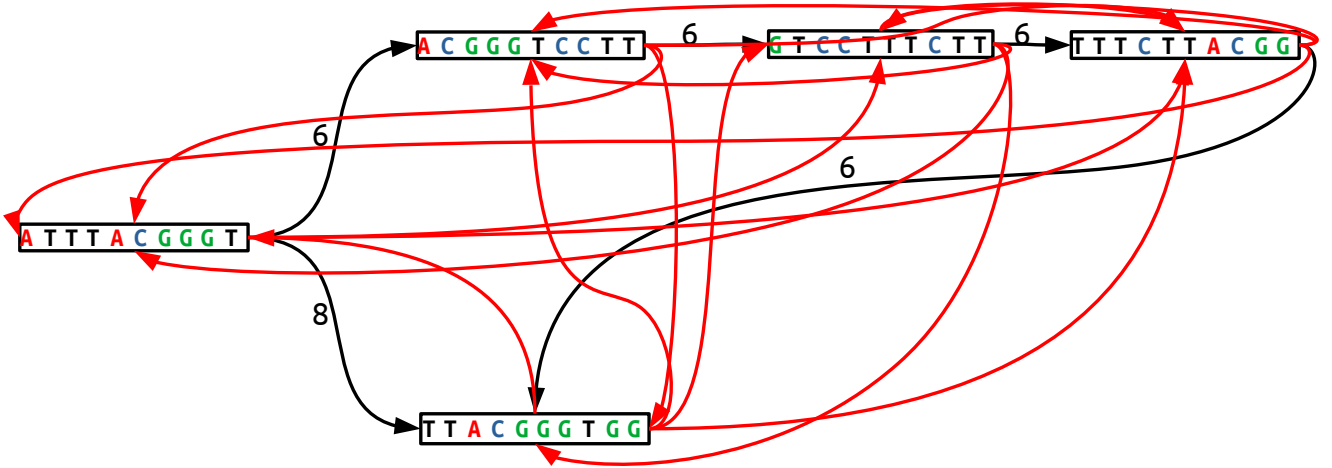
Assembly concession number 1: output fragments

In the real world, assemblers often provide pieces of genomes rather than complete ones

- **Assembly concession number 1: output fragments (ii)**

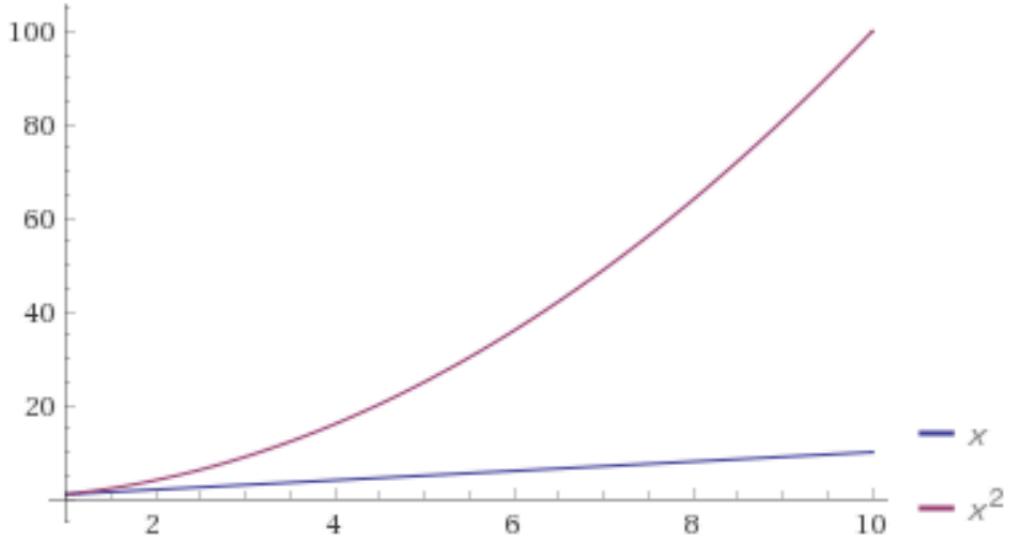


• Overlap graph prerequisite : all overlaps



• Overlap graph burden: number of reads

$\frac{n(n-1)}{2} = O(n^2)$ possible overlaps for n reads



Linear: 2X data 2X time

Overlap graph burden: number of reads (ii)

Quadratic: 2X data 4X time

• Overlap graph burden: number of reads

$\frac{n(n-1)}{2} = O(n^2)$ possible overlaps for n reads

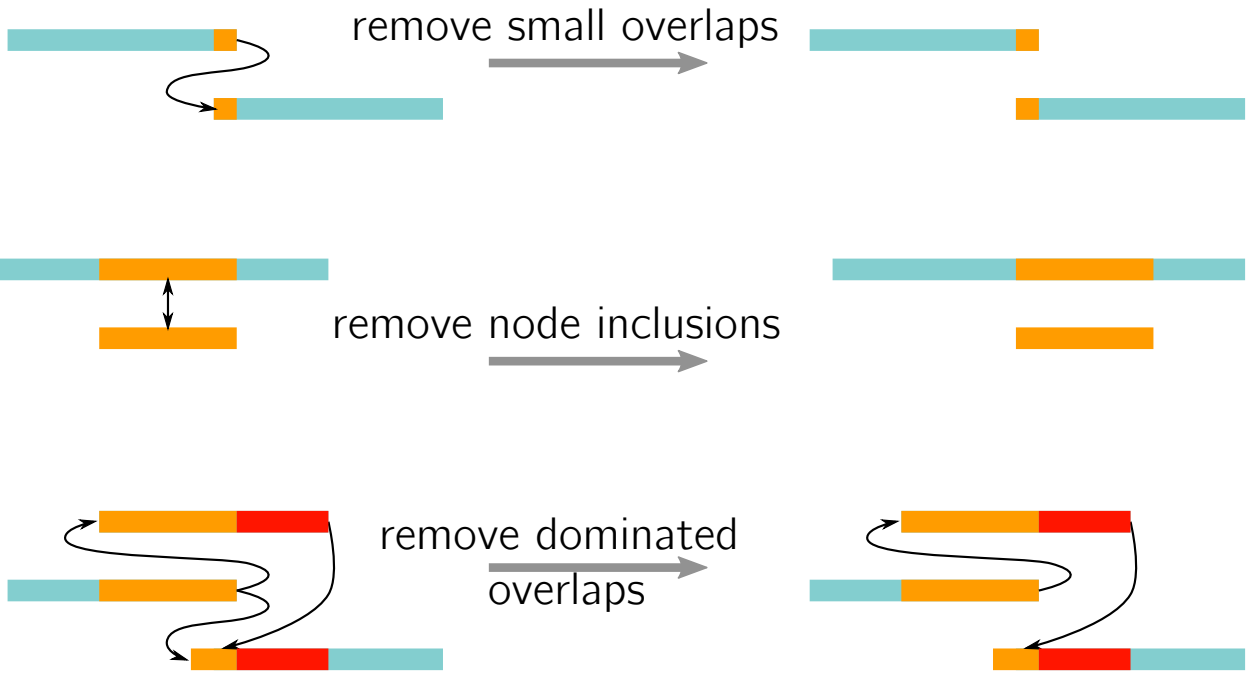
# Reads	# Overlaps
1000	499,500
10,000	50 million
100,000	5 billion
1 million	500 billion
10 million	50 trillion…

The overlap computation is not linear

Talking about CPU years on large genomes…

We have to be efficient here and focus on “relevant” overlaps

• Overlap graph simplifications



Overlap graphs in a nutshell

- Graphs of overlaps between the reads
- Can provide a global solution for assembly
- Can be difficult in real cases because it requires a lot of computation (overlaps)

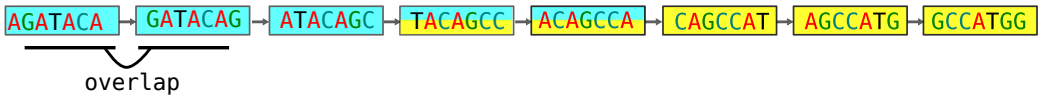
S. cerevisiae, *D. melanogaster*, human could be assembled using overlap graphs approaches (Celera (Myers et al. 2000), SGA (Simpson & Durbin 2011), ...)

• History: the introduction of another graph structure for assembly, according to MidJourney

Overlapping reads

AGATACAGCCA
TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATACAGCCATGG

- **Assembly idea number 3: Focus on genome words**

a genome: a succession of words

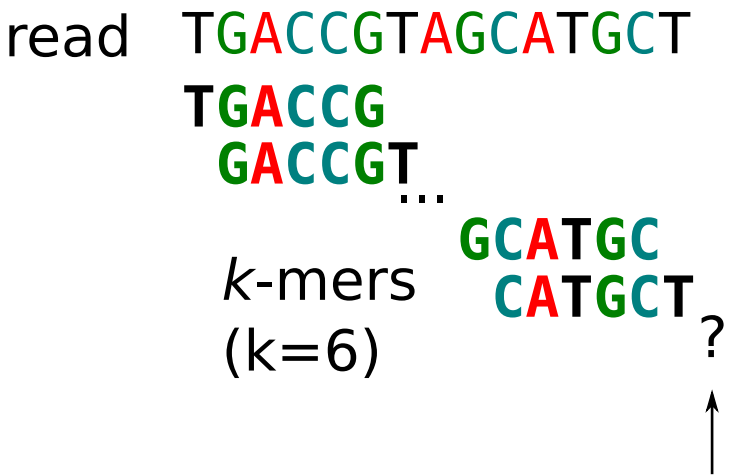
AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

GCATGC

CATGCT

- **Find consecutive genome words in read words**

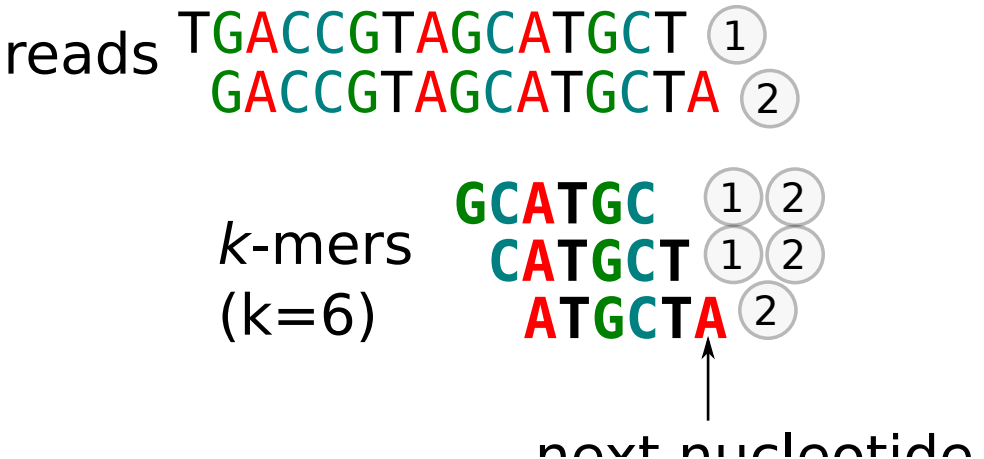
AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC



- **How to connect read words?**

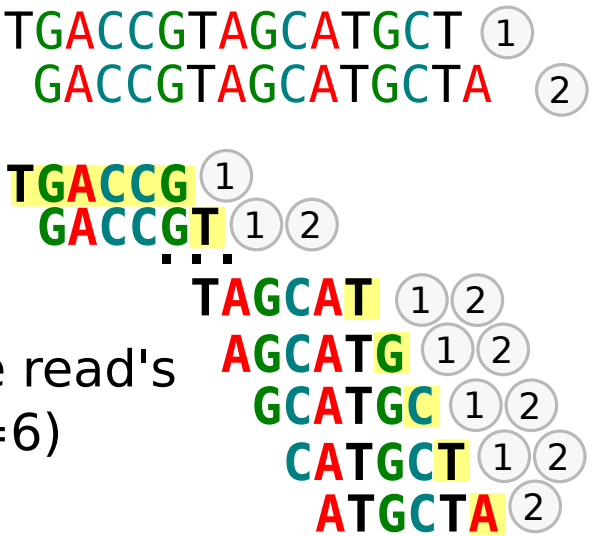
genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC



• Reconstitute larger genomic words

reads



extract the read's k-mers (k=6)

The de Bruijn graph

De Bruijn graph

Kmer=node

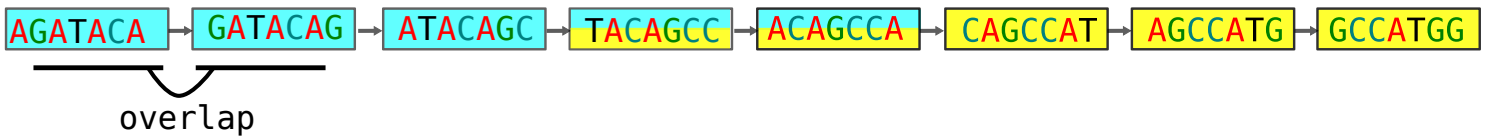
AGATACA + G + C + C + A

• de Bruijn graph assembly

Overlapping reads

AGATACAGCCA
TACAGCCATGG

De Bruijn graph



Resulting sequence

AGATACAGCCATGG

• de Bruijn graph time!

Reads

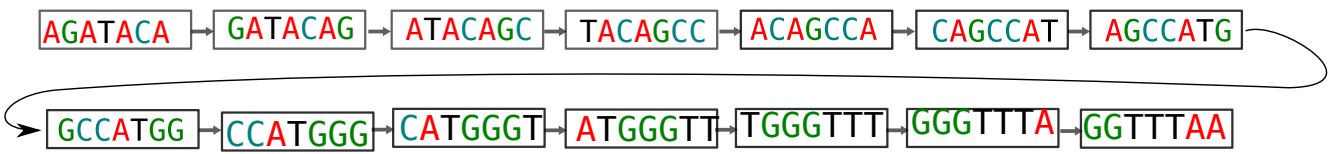
GCCATGGGTTT
TACAGCCATGG
AGCCATGGGTT
GCCATGGGTTT
AGATACAGCCA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
CAGCCATGCCT

Hint: Use 7-mers

• Solution

```
GATACAGCCATG
   TACAGCCATGG
    ACAGCCATGGG
    ACAGCCATGGG
     CAGCCATGGGT
      AGCCATGGGTT
       GCCATGGGTTT
       GCCATGGGTTT
          CATGGGTTTAA
           CATGGGTTTAA
```

De Bruijn graph



• de Bruijn graphs abstract redundancy

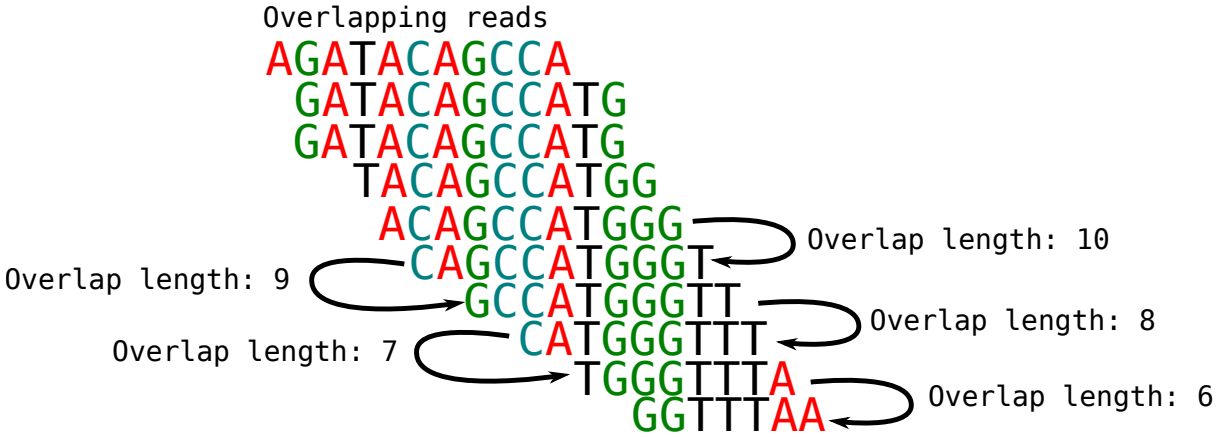
```
GATACAGCCATG
GATACAGCCATG
  TACAGCCATGG
   ACAGCCATGGG
   ACAGCCATGGG
    CAGCCATGGGT
     AGCCATGGGTT
      GCCATGGGTTT
      GCCATGGGTTT
        CATGGGTTTAA
        CATGGGTTTAA
```

62 **(non distinct)** 7-mers in the reads

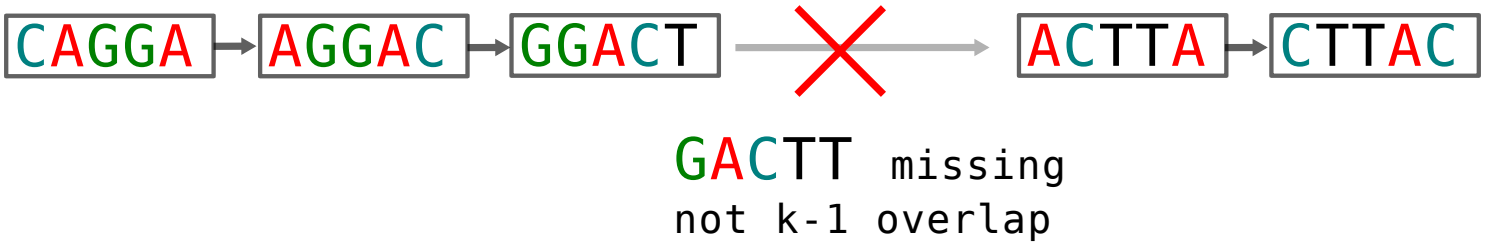
De Bruijn graph

14 **distinct** 7-mers in the De Bruijn graph

• de Bruijn graphs only rely on $k-1$ overlaps



• de Bruijn graphs limitation 1: Fixed overlaps



GGACT and ACTTA overlap is only of size 3 !

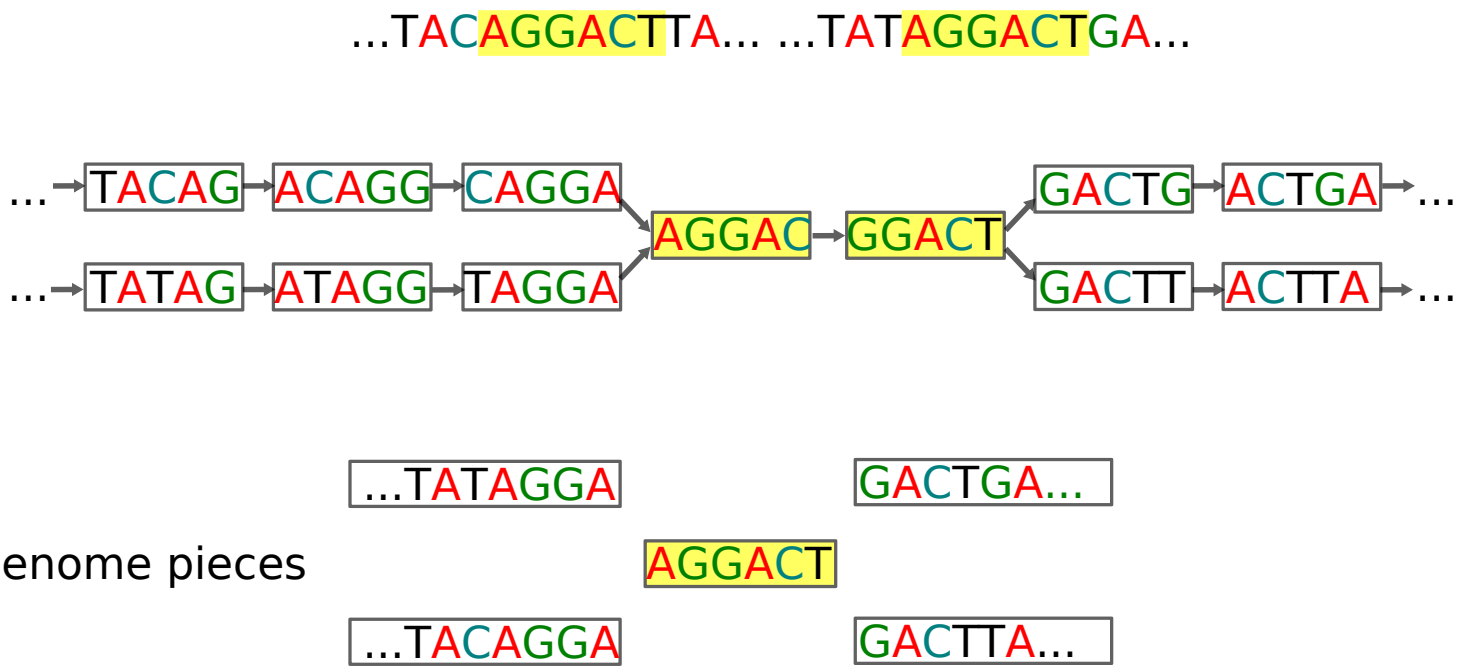
• de Bruijn graphs limitation 2: Repeats

…TACAGGACTTA… …TATAGGACTGA…

#v(0.4cm)

each k-mer appears only once in a de Bruijn graph

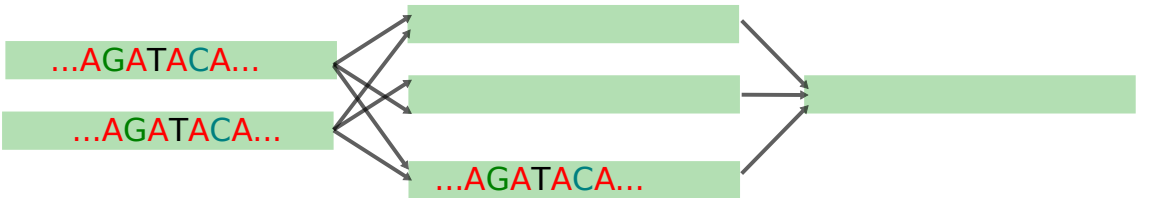
- **de Bruijn graph limitation**



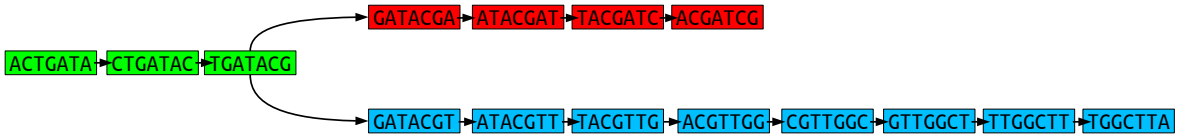
de Bruijn graph limitation (ii)

• de Bruijn graph versus overlap graph

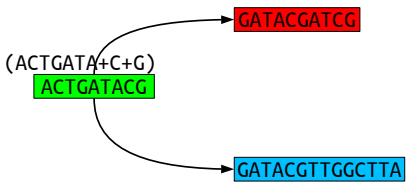
Overlap graph from the reads



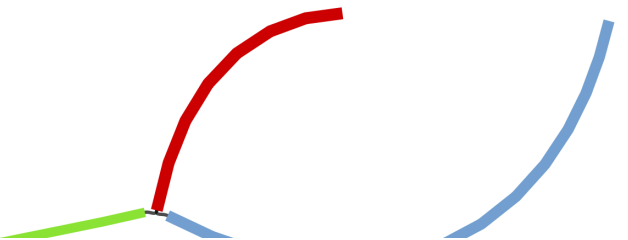
On the representation of de Bruijn graphs



Compacted De Bruijn graph:

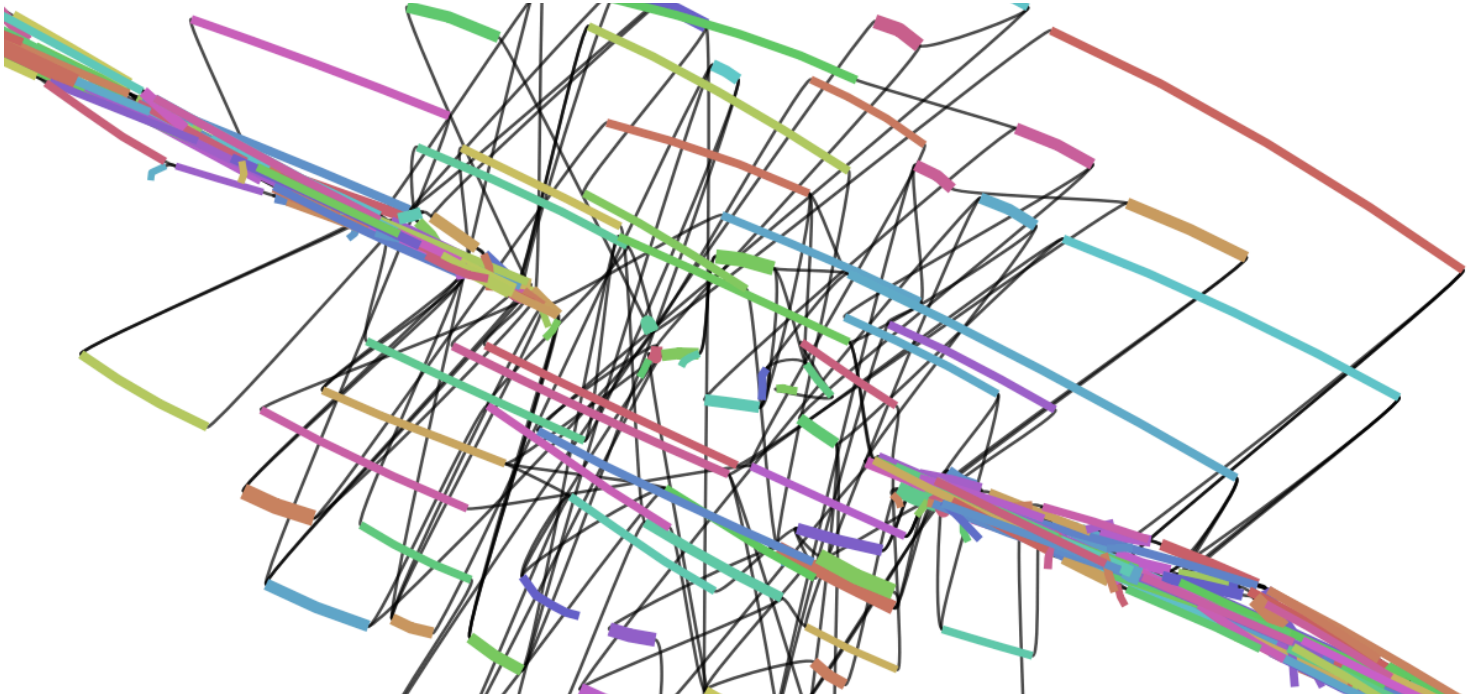


Graphical representation
(.gfa plot using Bandage):



de Bruijn graph on a real dataset

• de Bruijn graph on a real dataset ZOOMED IN

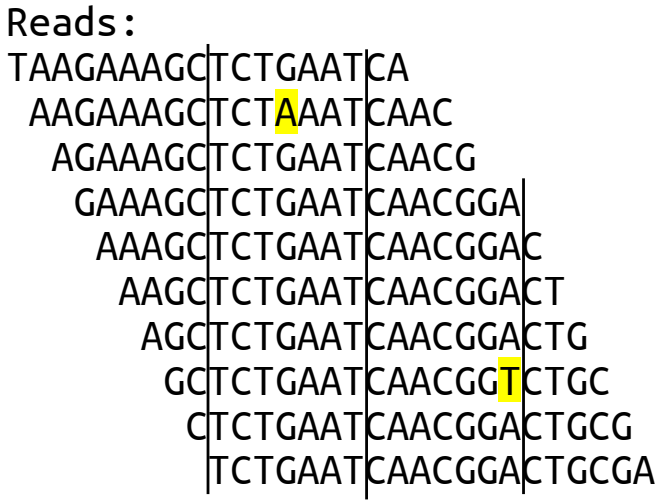


• Sequencing errors

```
Reads:
 ATCGCTATCG
    GGTTTCGTTA
       ATCGATACGG

TCGCTA
GGTTTC
 ATCGAT
...
Are not genomic kmers...
```

- **Erroneous k-mers vs genomic k-mers**

TAAGAAAGCTCTGAATCAACGGACTGCGACA



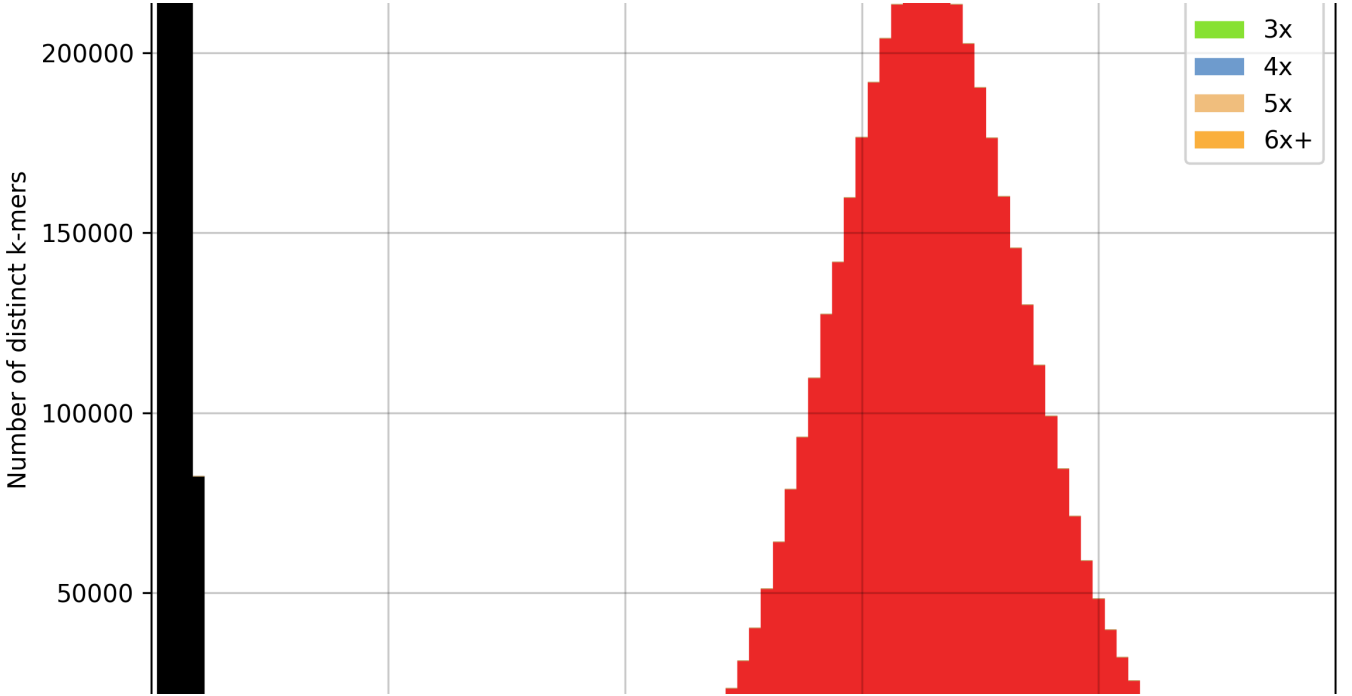
9 times TCTGAAT
1 time TCTAAAT

6 times CAACGGA
1 time CAACGGT

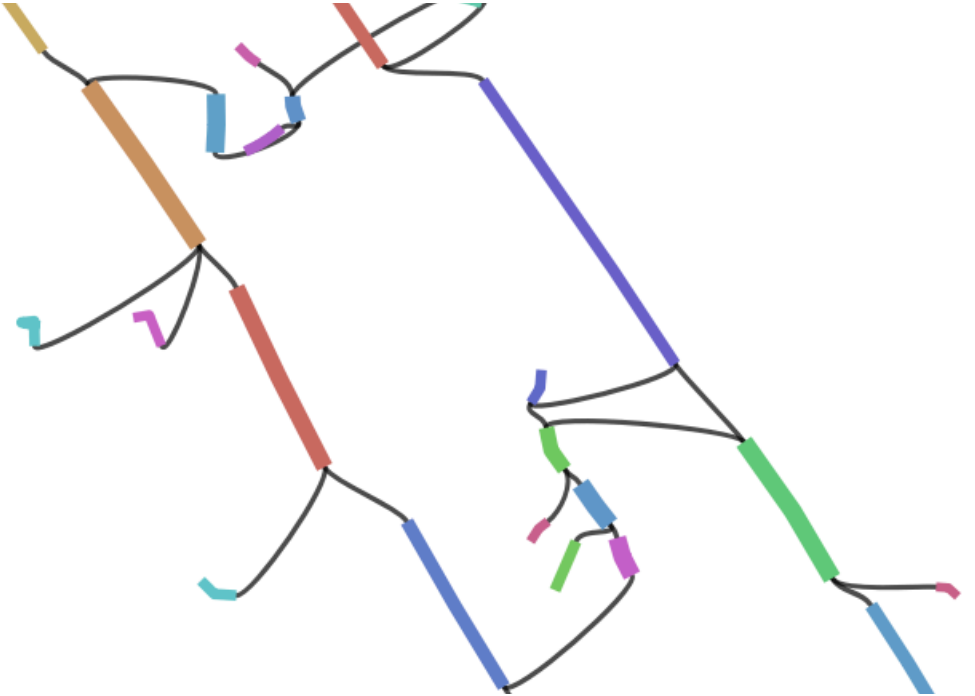
Erroneous k-mers vs genomic k-mers (ii)

Erroneous k-mers are seen less than genomic ones

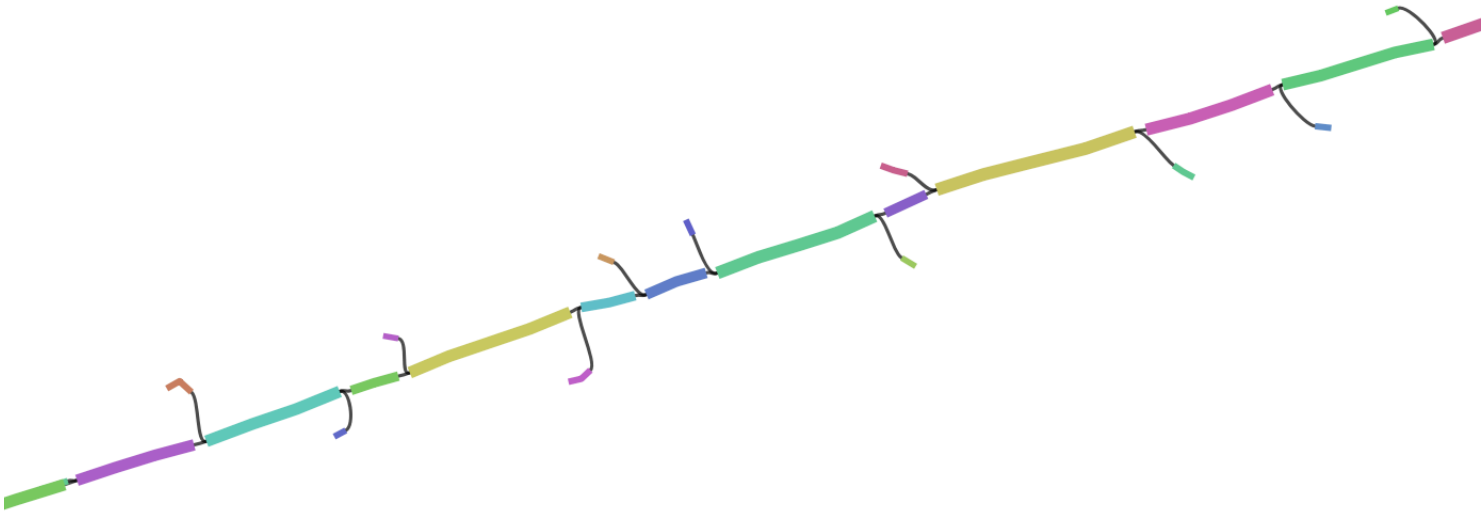
K-mer histogram



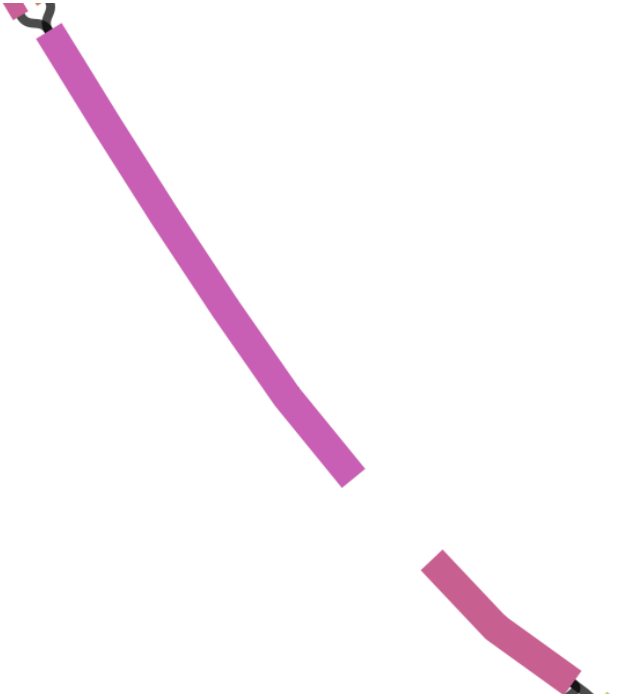
• Removing unique k-mers



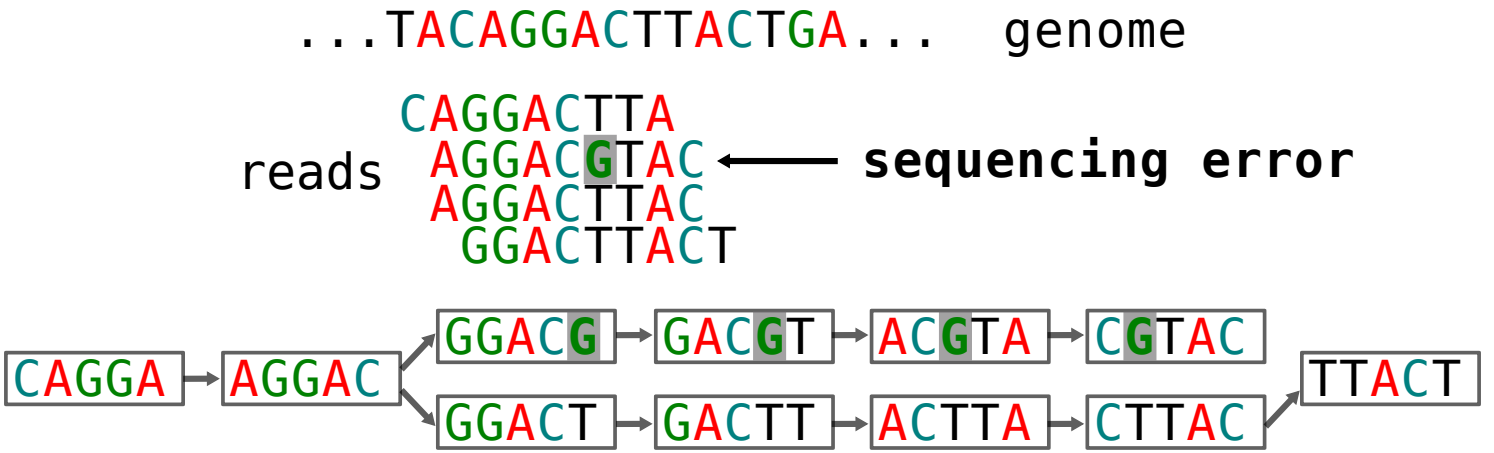
• Removing k-mers seen less than 3 times



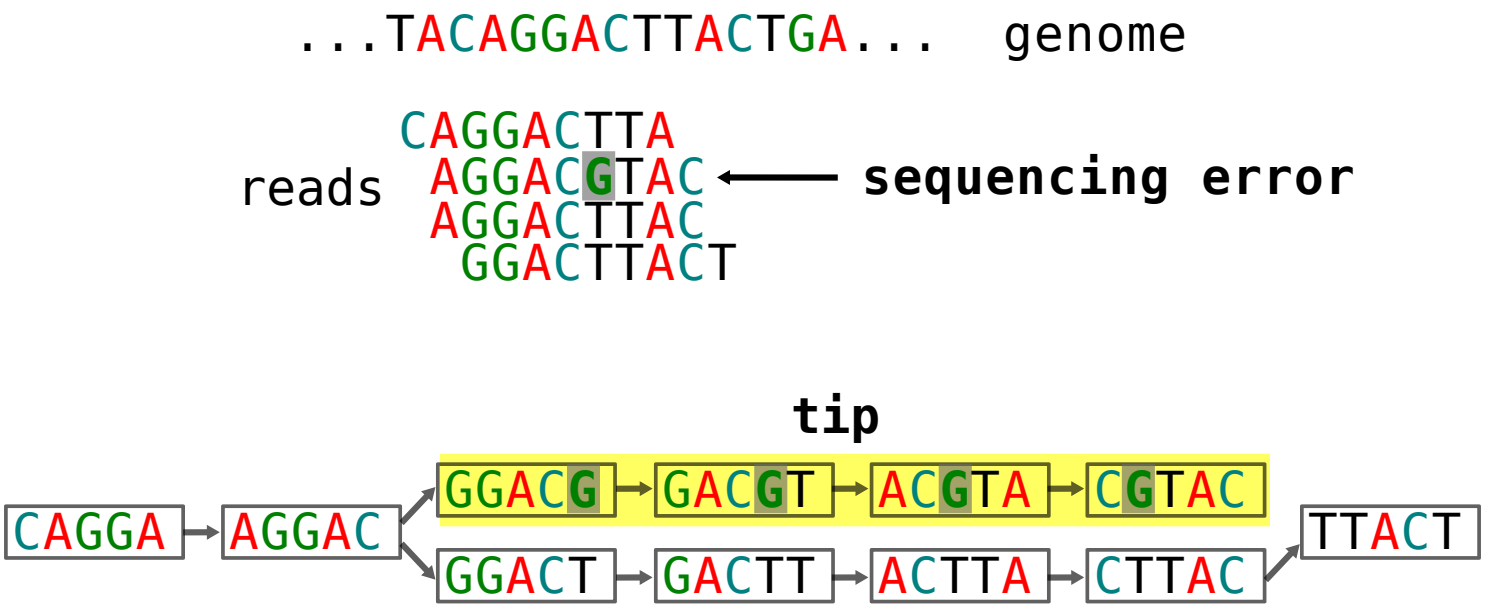
- **Removing k-mers seen less than 4 times**



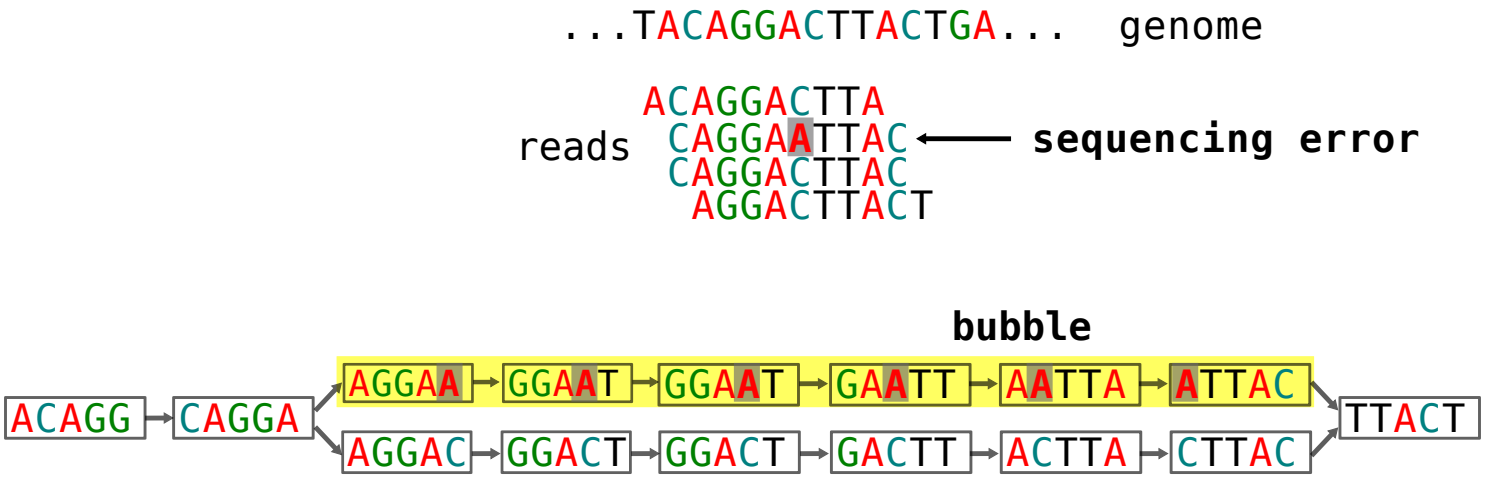
• Errors in de Bruijn graphs



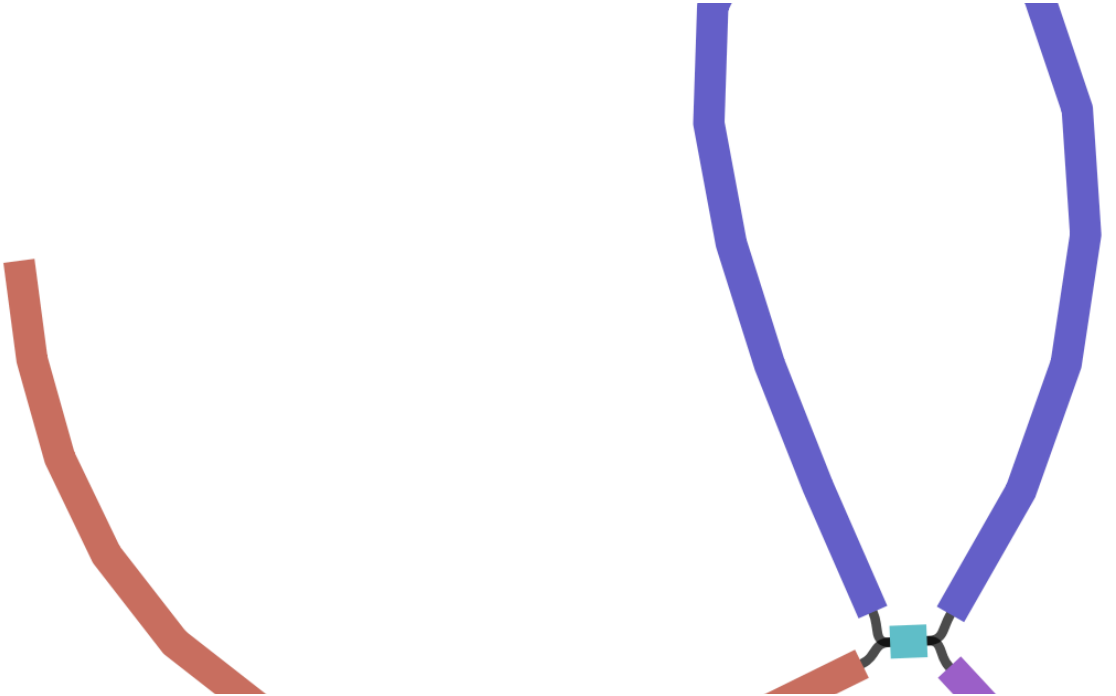
• Errors in de Bruijn graphs



• Errors in de Bruijn graphs



- (Almost assembled phage !)



de Bruijn graphs in a nutshell

- Graph of words of size k, k–1 overlaps
- Collapses identical k-mers
- Very successful, have replaced the overlap graphs with high throughput sequencing data
- Still outputs fragments of the genome

White spruce, 20 gigabases

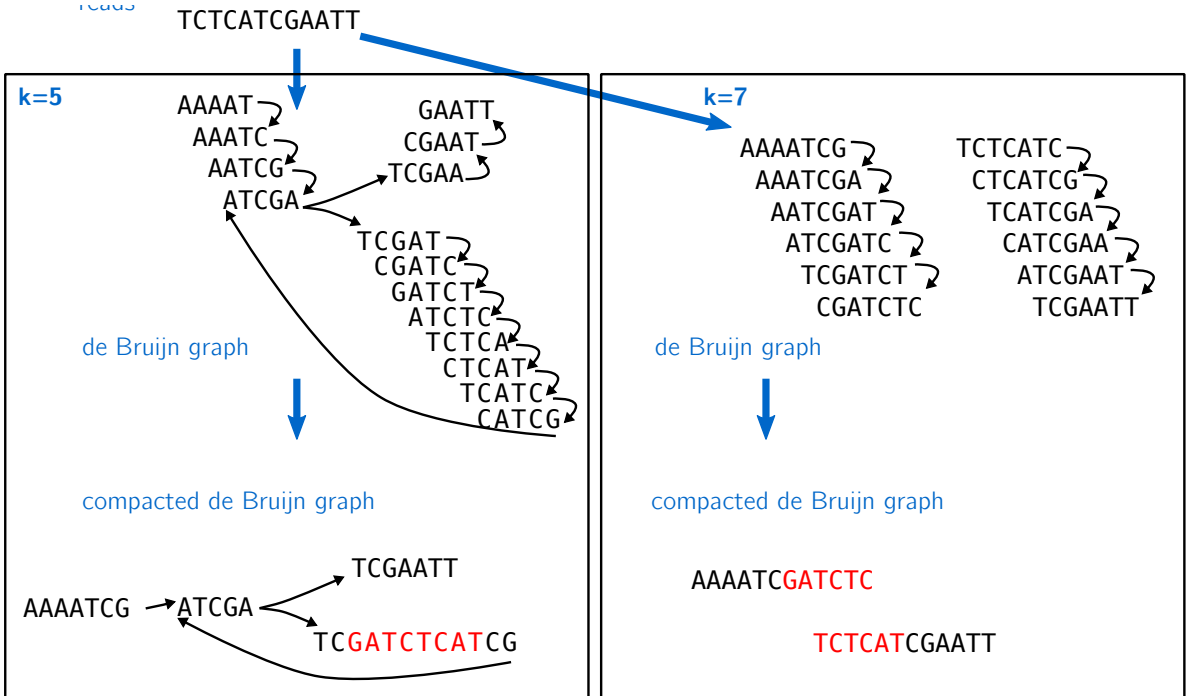
• Multiple k assembly

Most de Bruijn graph assemblers can now perform several assemblies with different k-mer sizes to produce an improved "super" assembly

Exercice

body

• Multiple k assembly



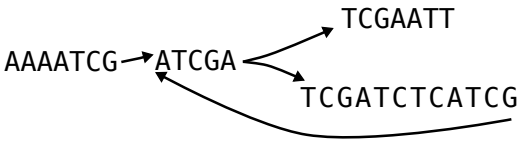
• Multiple k assembly (ii)

We are missing GATCTCA and ATCTCAT in the second graph.

But they are present in the first graph!

• Multiple k assembly

compacted de Bruijn graph
k=5



k-mers k=7
from cdBG k=5

AAAATCG
AAATCGA
TCGATCT
CGATCTC
GATCTCA
TCGAATT
ATCTCAT
TCTCATC
CTCATCG

+

k-mers k=7
from the reads

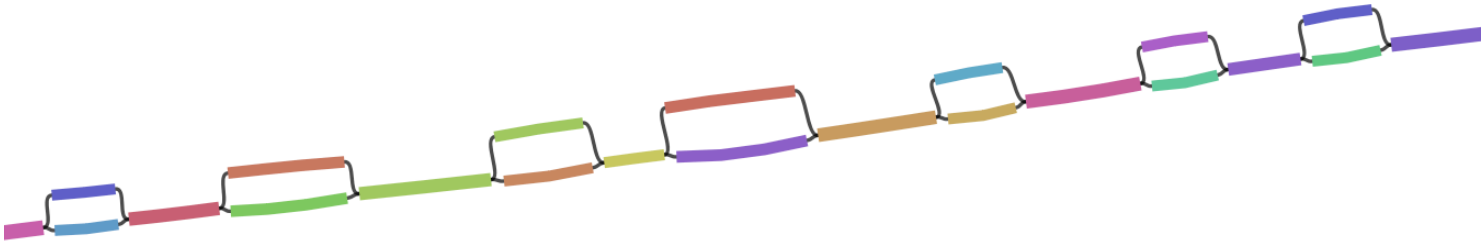
AAAATCG TCTCATC
AAATCGA CTCATCG
AATCGAT TCATCGA
ATCGATC CATCGAA
TCGATCT ATCGAAT
CGATCTC TCGAATT

→

compacted de Bruijn graph
k=7

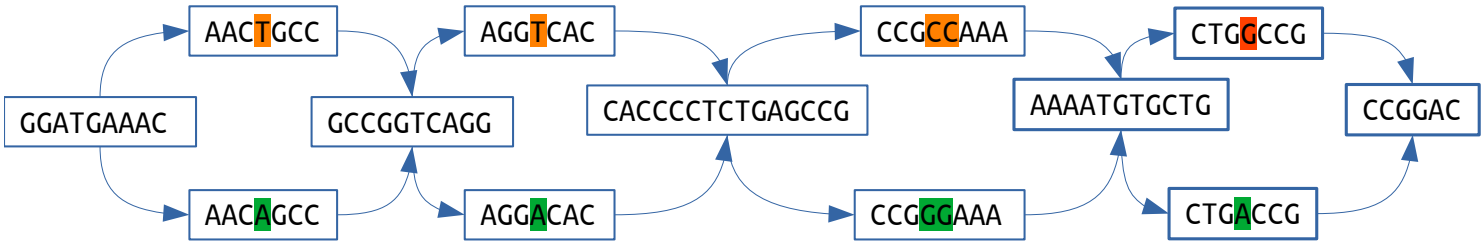
AAAATCGATCTCATCGAATT

de Bruijn graph on an eukaryota



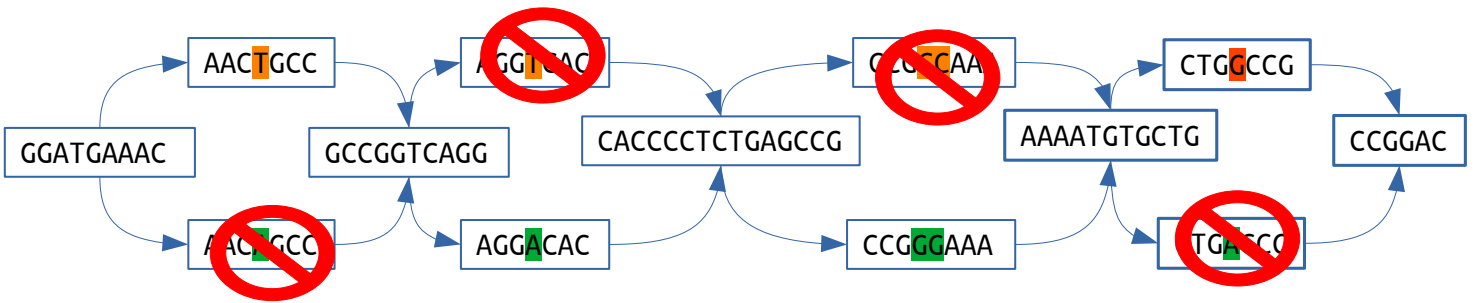
- **Two or more genomes per individual**

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



- **Two or more genomes per individual**

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC



Assembly:

• Assembly concession number 2: collapse variability

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

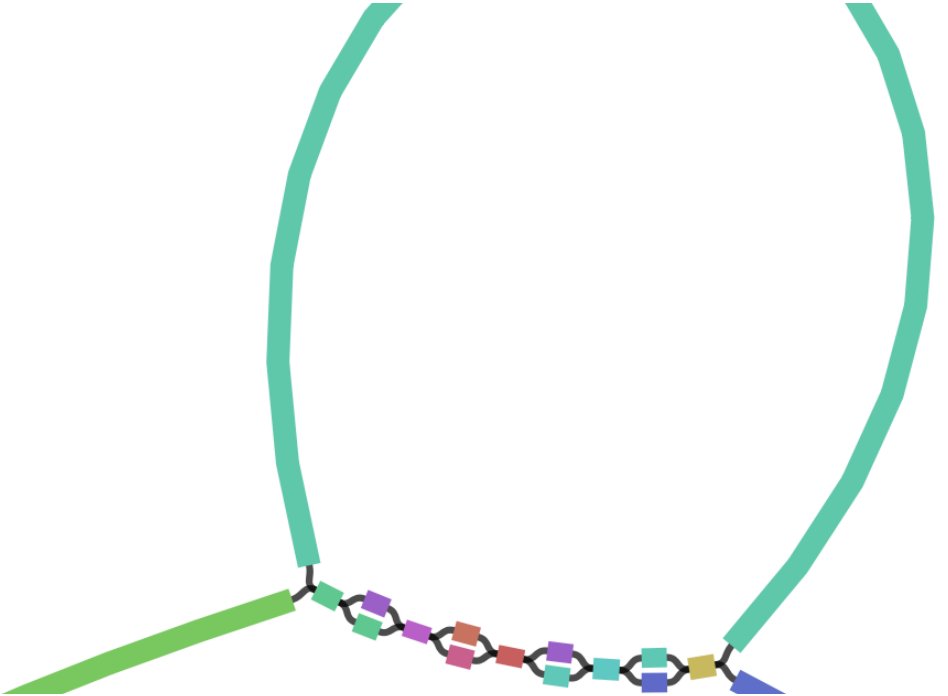
Assembly:

GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGGCCGGAC

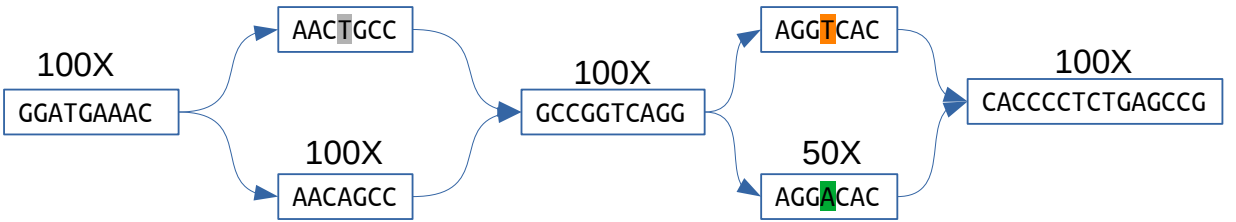
Reads:

GATGAAACTG
ATGAAACAGC
TGAAACAGCCG
GAAACTGCCGG
AAACTGCCGGT
AACAGCCGGTC
ACAGCCGGTCA
CTGCCGGTCAG

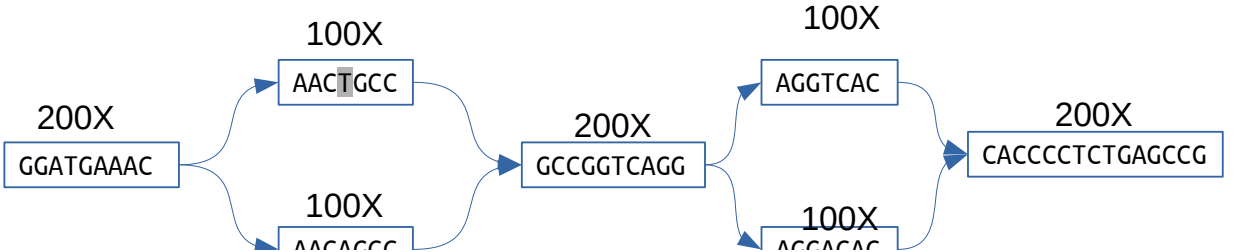
• Paralog genes/repeats



• Paralog genes/repeats in graph



Quasi repeats



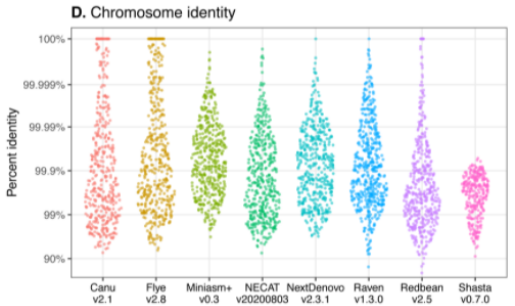
• An assembler is a set of heuristics

Graph cleaning heuristics

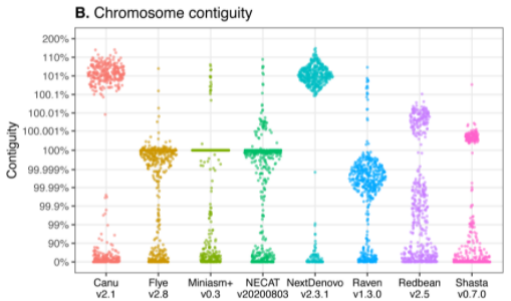
body

An assembly is a model

1. Assemblies contain errors
2. Different tools can produce very similar assemblies
3. A single tool can produce very different assemblies with small changes of parameters(!)



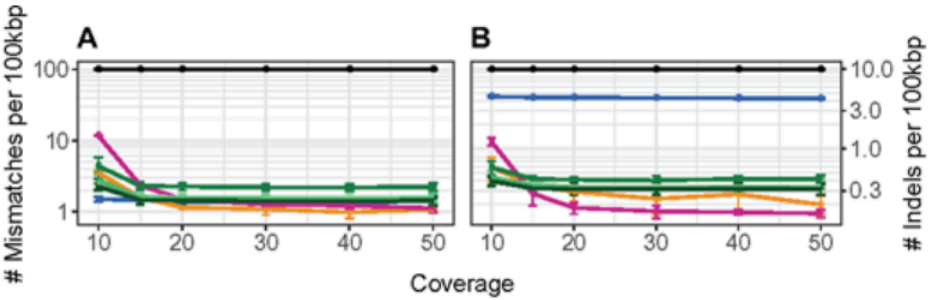
An assembly is a model (ii)



From https://github.com/rrwick/Long-read-assembler-comparison

What do we do post-assembly?

1. Assess its quality
2. Improve it
3. Use it!



Two ways to polish an assembly according to MidJourney

Evaluate assembly according to a reference

Contigs can be mapped and compared to a reference/closely related genome

• Evaluate assembly according to a reference (ii)

• Evaluate assembly according to a reference (iii)

From https://quast.bioinf.spbau.ru/manual.html

Assembly statistics

Total aligned length	4 776 214	4 568 317	4 553 809	4 550 150
NGA50	69 801	122 647	133 309	112 446
LGA50	21	14	12	14
Misassemblies				
# misassemblies	4	0	0	4
Misassembled contigs length	231 767	0	0	435 515
Per base quality				
# mismatches per 100 kbp	2.09	2.69	1.03	3.19
# indels per 100 kbp	0.57	1.31	0.29	1.98
# N's per 100 kbp	24.59	0	17.55	94.19
Statistics without reference				
# contigs	176	95	92	90
Largest contig	248 481	235 933	285 196	264 944
Total length	4 777 853	4 571 292	4 557 363	4 552 266
Total length (>= 1000 bp)	4 757 929	4 562 458	4 548 710	4 544 453
Total length (>= 10000 bp)	4 562 801	4 478 614	4 466 223	4 475 223
Total length (>= 50000 bp)	3 248 113	3 833 793	3 812 315	3 817 904
BUSCO completeness				

Assembly statistics (ii)

From http://cab.cc.spbu.ru/quast/

• Assembly continuity

N50

body

• Assembly continuity

N50

body

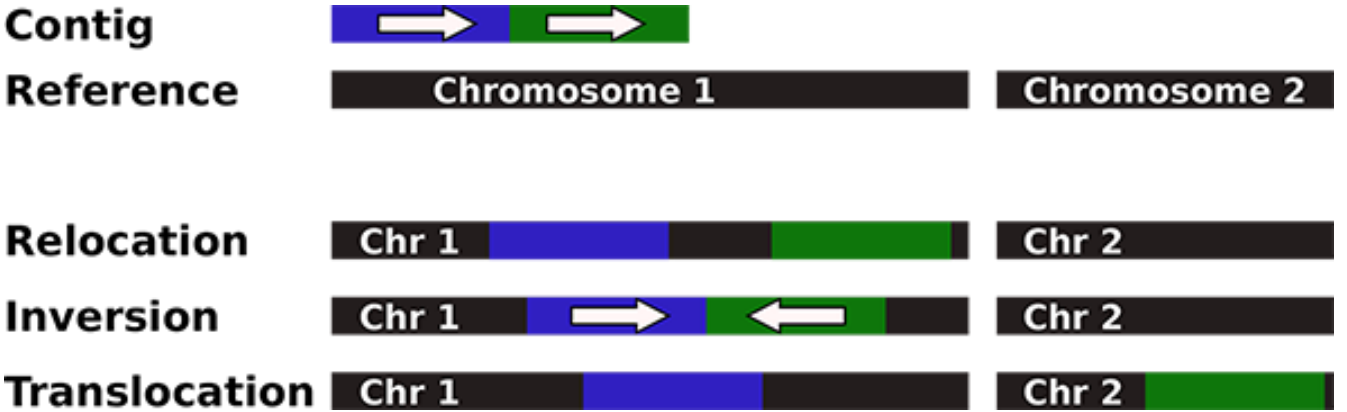
N75

body

NGA50

body

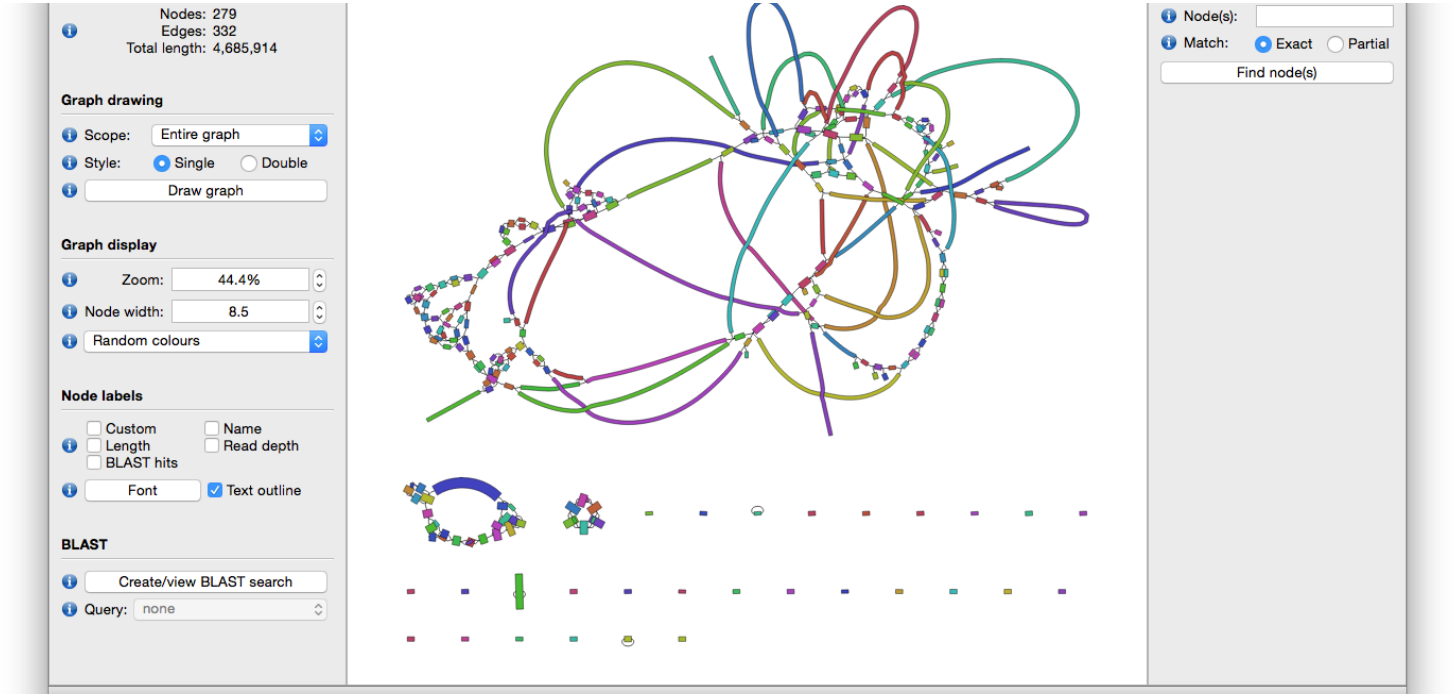
• Misassemblies



• Visualize assembly

Bandage tool can visualize assembly graphs (GFA)

• Visualize assembly (ii)



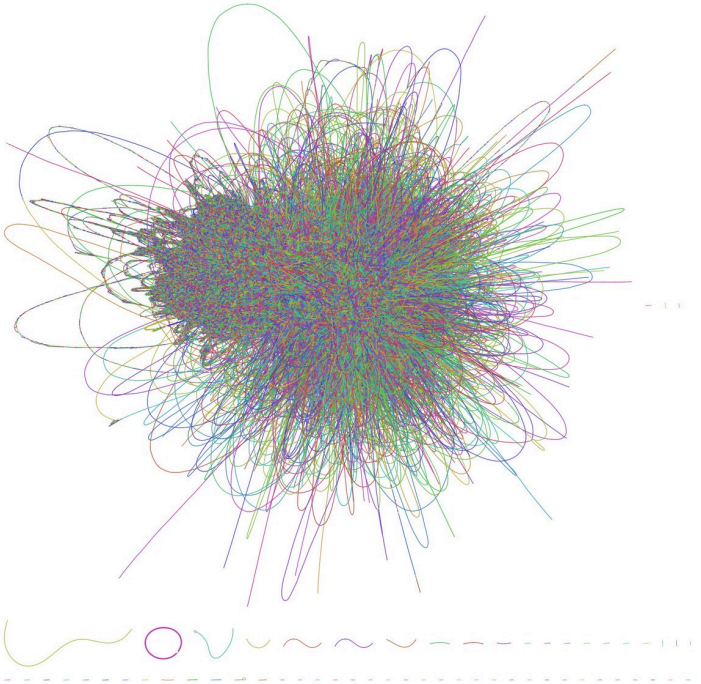
• Visualize assembly (iii)

From https://rwick.github.io/Bandage

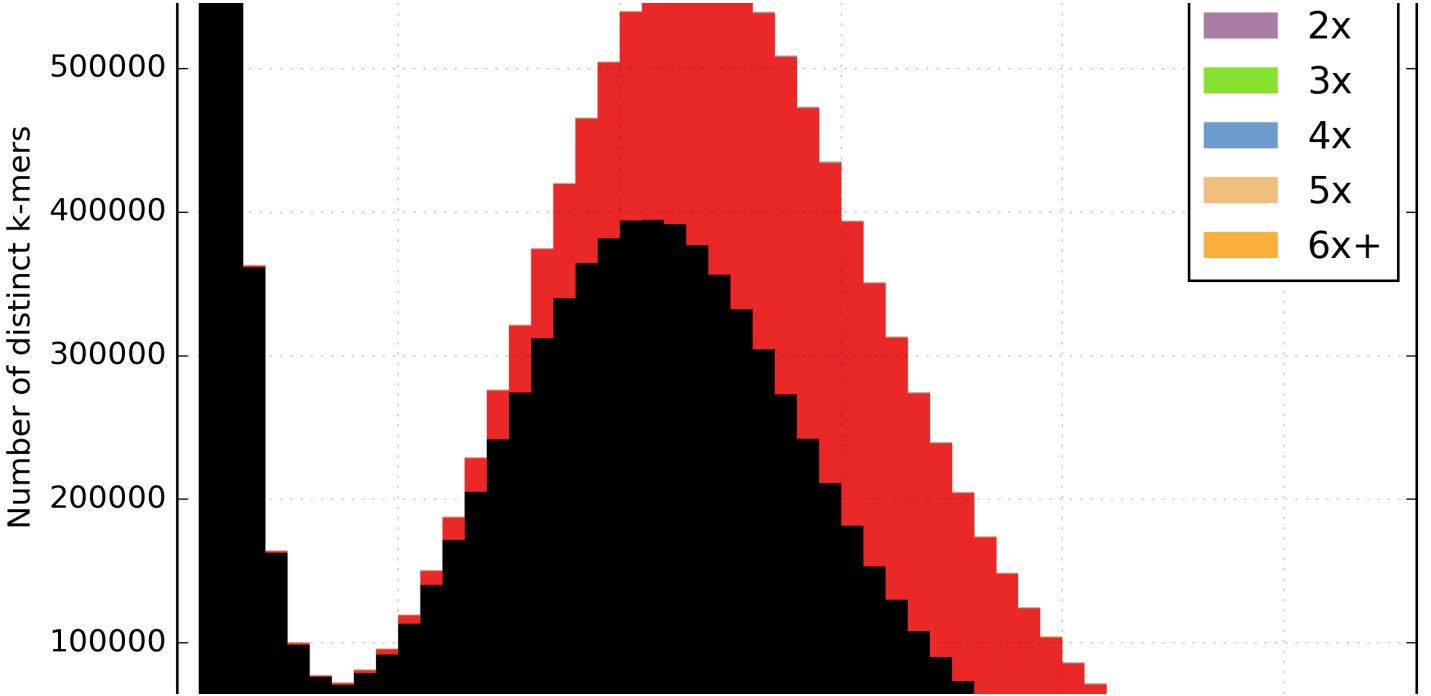
• Visualize assembly

Bandage tool can visualize assembly graphs (GFA)

• Visualize assembly (ii)



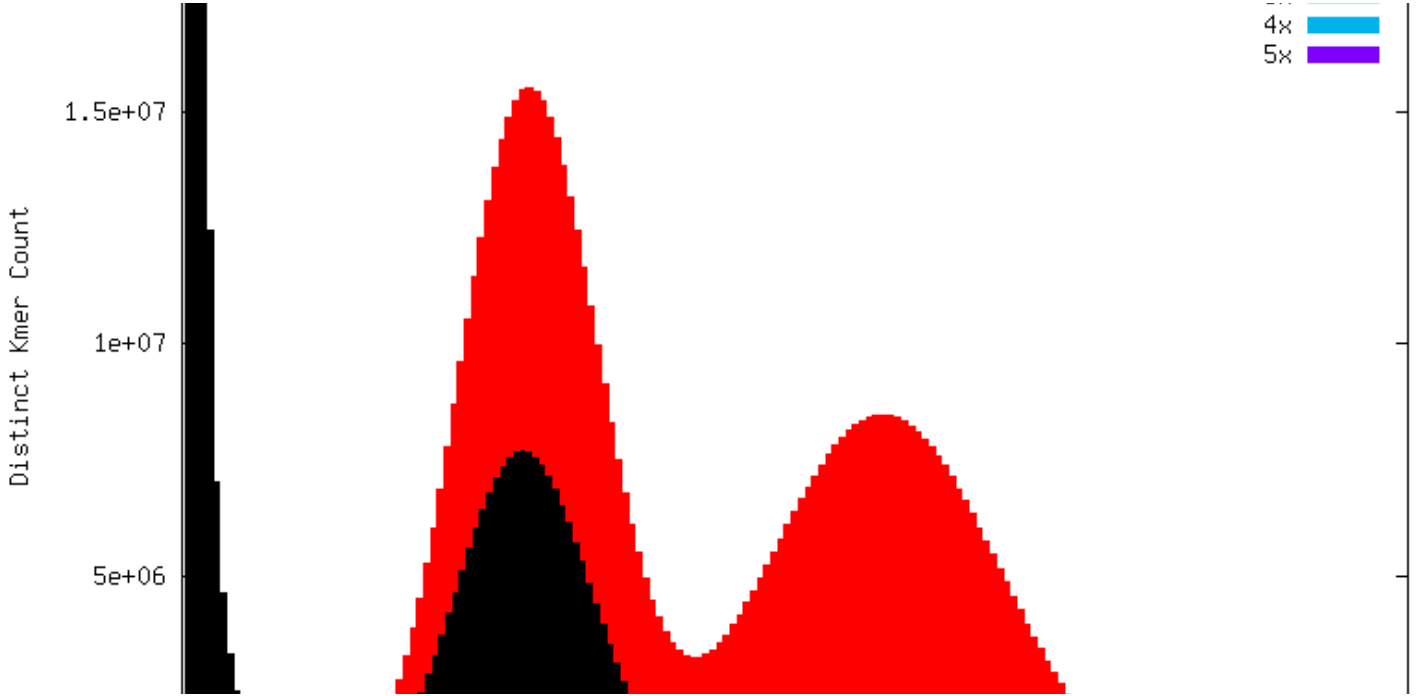
• K-mer spectrum visualization with KAT



• K-mer spectrum visualization with KAT (ii)

From https://kat.readthedocs.io/en/latest/

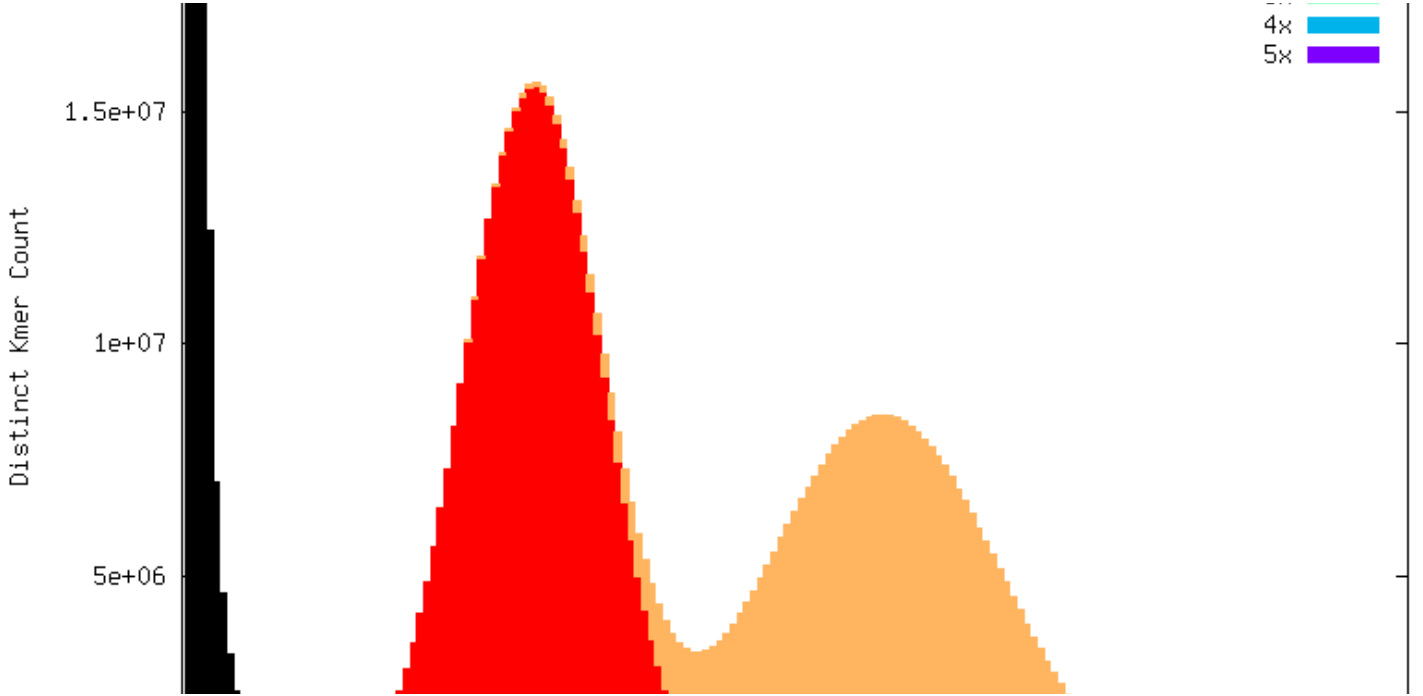
• K-mer spectrum visualization with KAT



• K-mer spectrum visualization with KAT (ii)

From https://kat.readthedocs.io/en/latest/

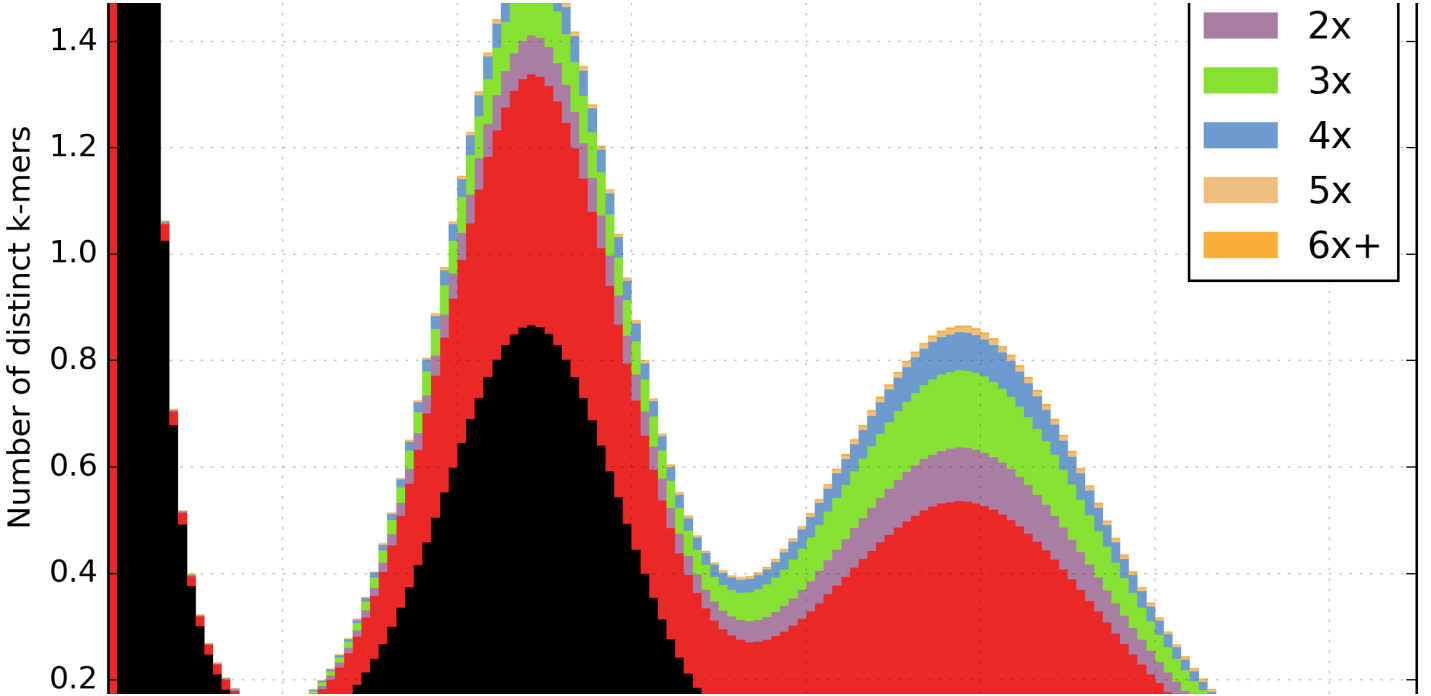
• K-mer spectrum visualization with KAT



• K-mer spectrum visualization with KAT (ii)

From https://kat.readthedocs.io/en/latest/

• K-mer spectrum visualization with KAT



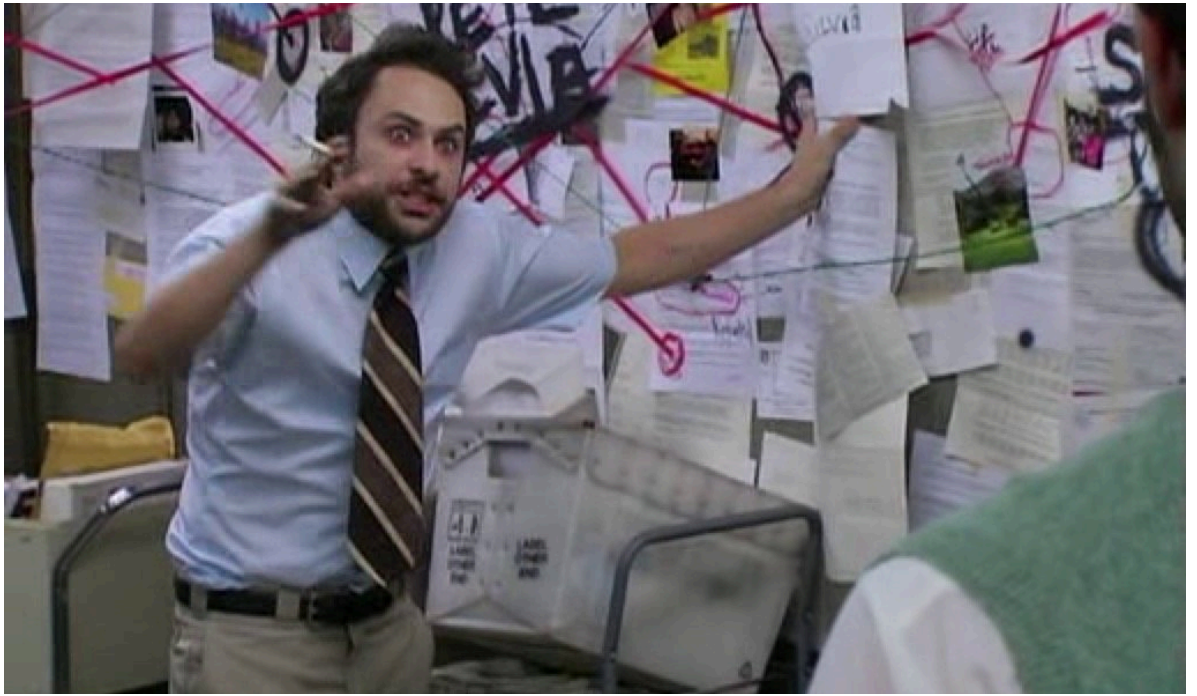
• K-mer spectrum visualization with KAT (ii)

From https://kat.readthedocs.io/en/latest/

The end

… of the theoretical part (or is it?)

The end (ii)



• Sanger

- Medium reads approx 1000 bp
- Very low error rate approx 0.01 %
- Low throughput (up to billion of reads per run)
- Costly ($500/Mb)

• Sanger (ii)

No longer used for assembly

• Second generation sequencing

NextSeq Series ✪

HiSeq 4000 System

HiSeq X Series‡

NovaSeq 6000 System

- Short reads approx 150 bp
- Low error rate $< 1\%$
- High throughput (up to billion of reads per run)
- Cheap ($0.50/Mb)
- GC bias

Which assembly strategy is best suited?

- Short reads approx 100 bp
- Low error rate below 1%
- High throughput (up to billions of reads per run) Based on long reads properties, which assembly solution would you choose and why?

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

Paradigm Breakdown

Greedy	Overlap graph	De Bruijn graph
body	body	body

State-of-the-art

Well performing assemblers

body

State-of-the-art (ii)

State-of-the-art (iii)

Other notable assemblers

body

• Third generation sequencing

• Third generation sequencing (ii)

- Long reads approx 10-100 kbp
- High error rate (up to

$$10\%$$

)

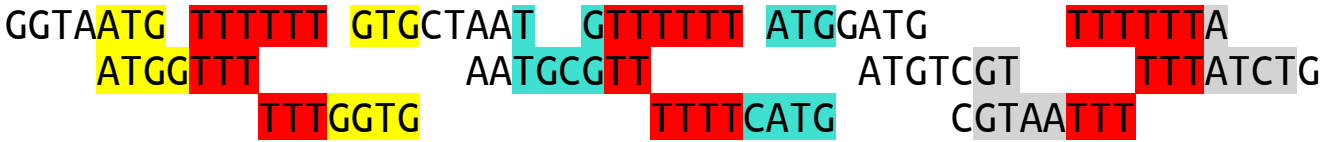
Third generation sequencing (iii)

- High throughput (up to millions of reads per run)

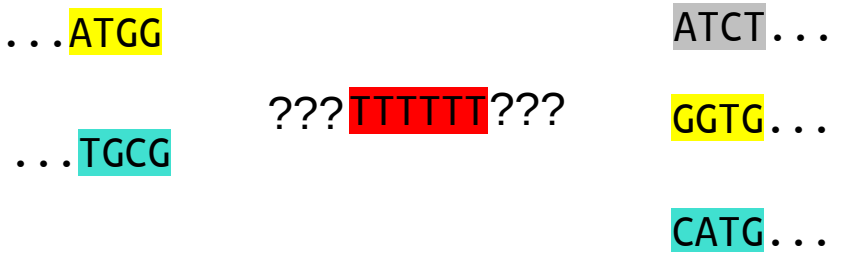
Nanopore VS Pacbio

Nanopore	Pacbio
body	body

- **Repeats spanning**

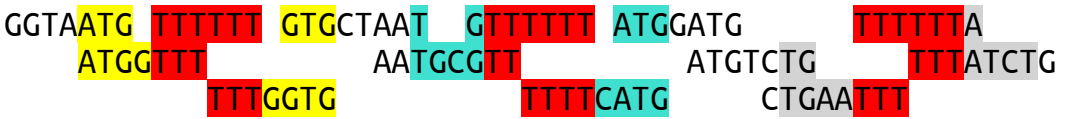


Contexts of the repeat:



Repeats spanning

Reads:



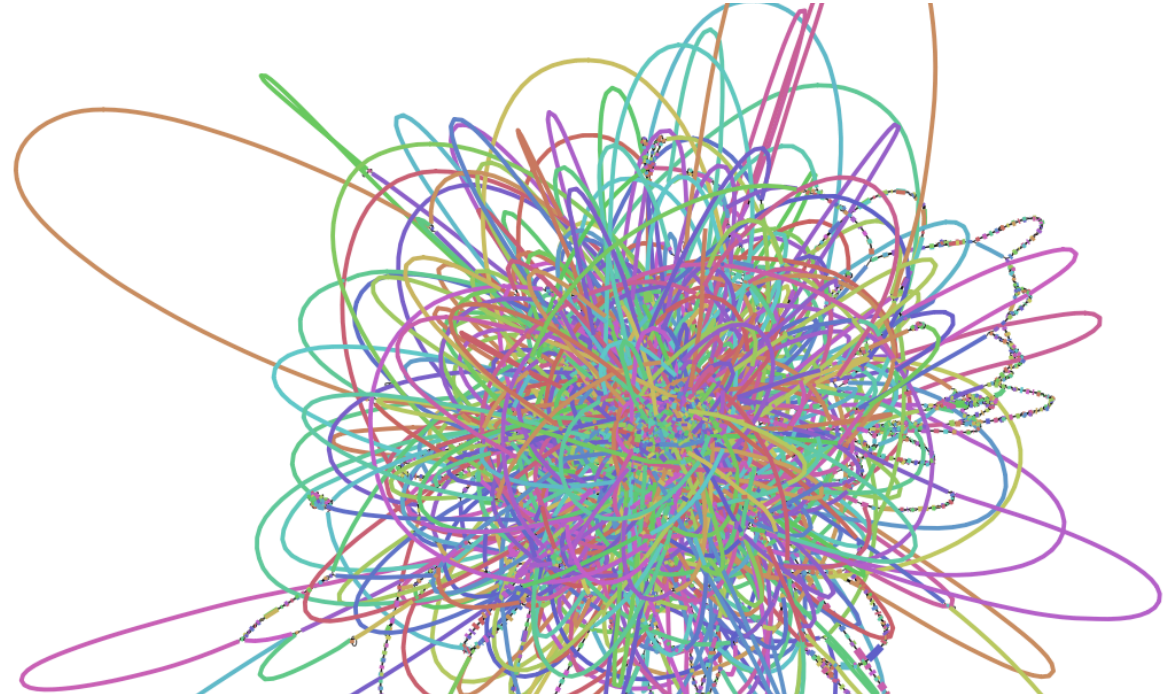
Long reads:

Contexts of the repeats:

• Read length matters

Read size=21

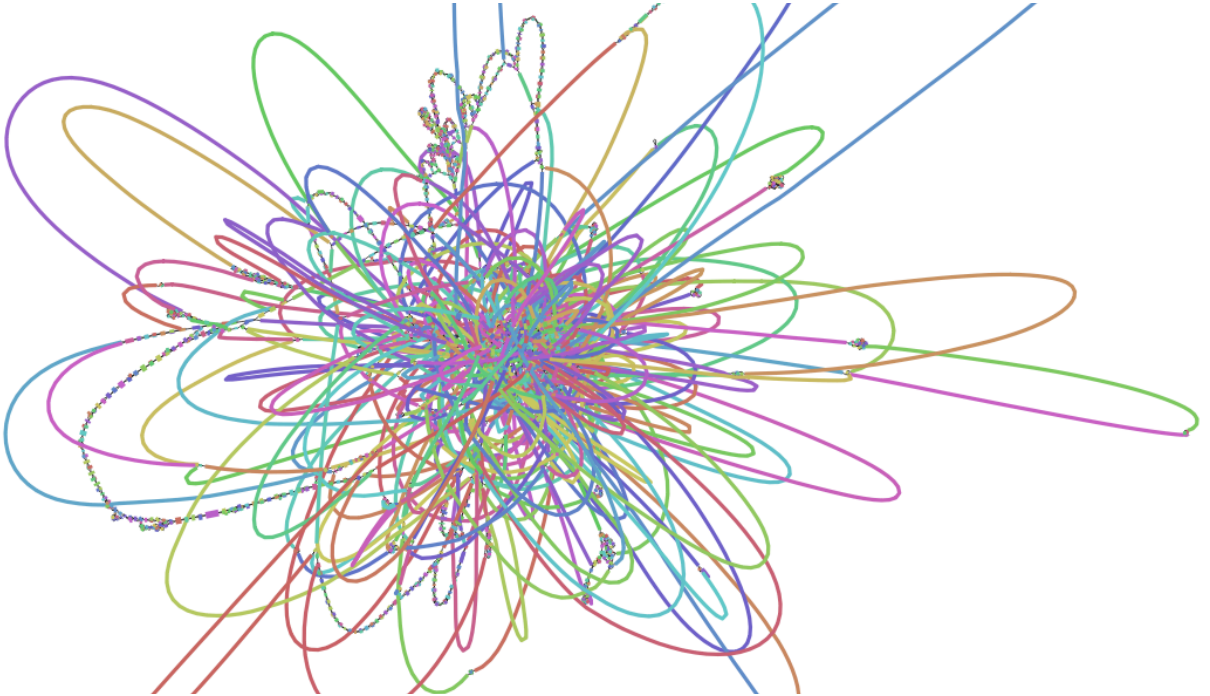
• Read length matters (ii)



• Read length matters

Read size=31

• Read length matters (ii)



• Read length matters

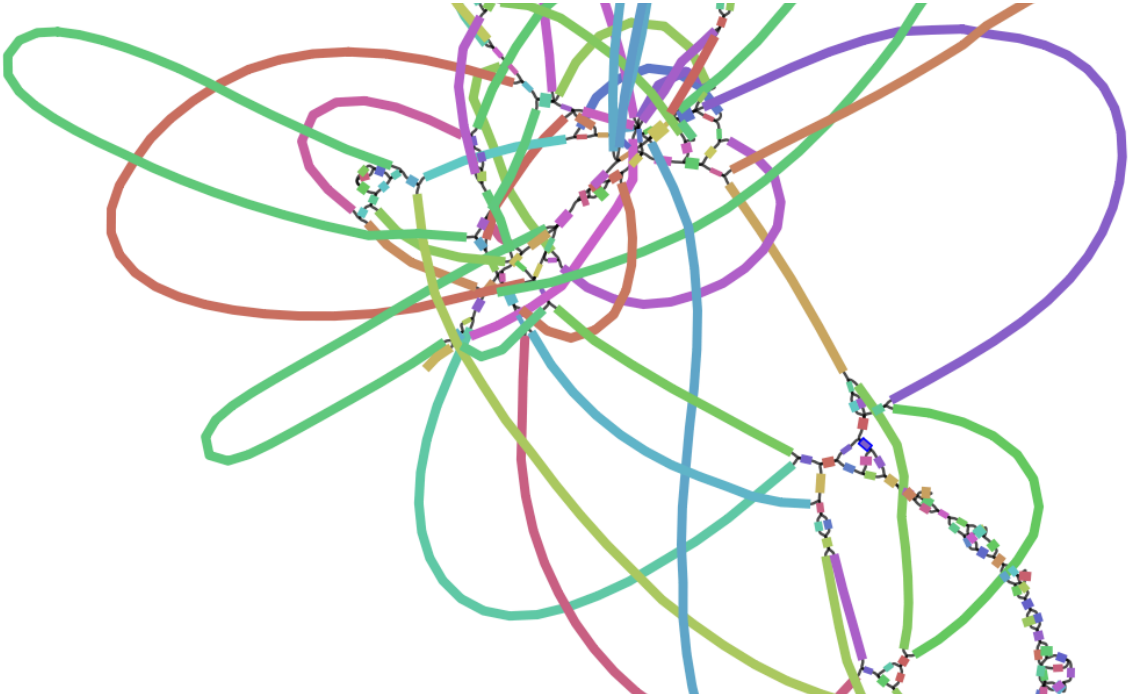
Read size=63

• Read length matters (ii)

• Read length matters

Read size=255

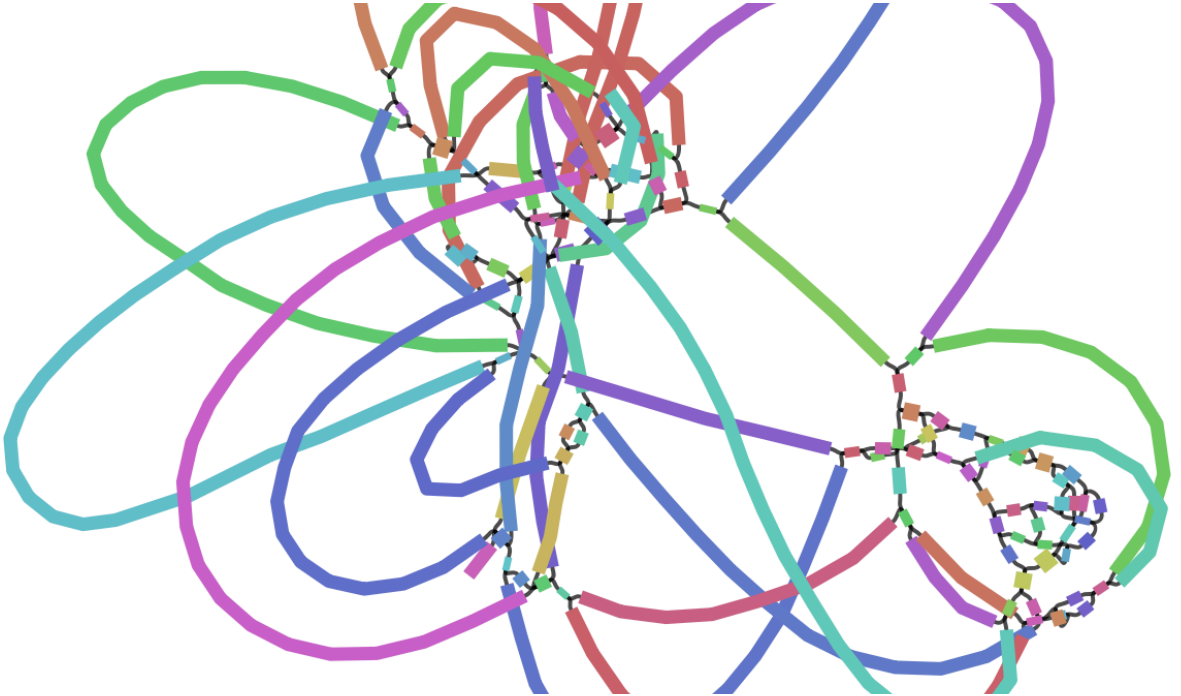
• Read length matters (ii)



• Read length matters

Read size=500

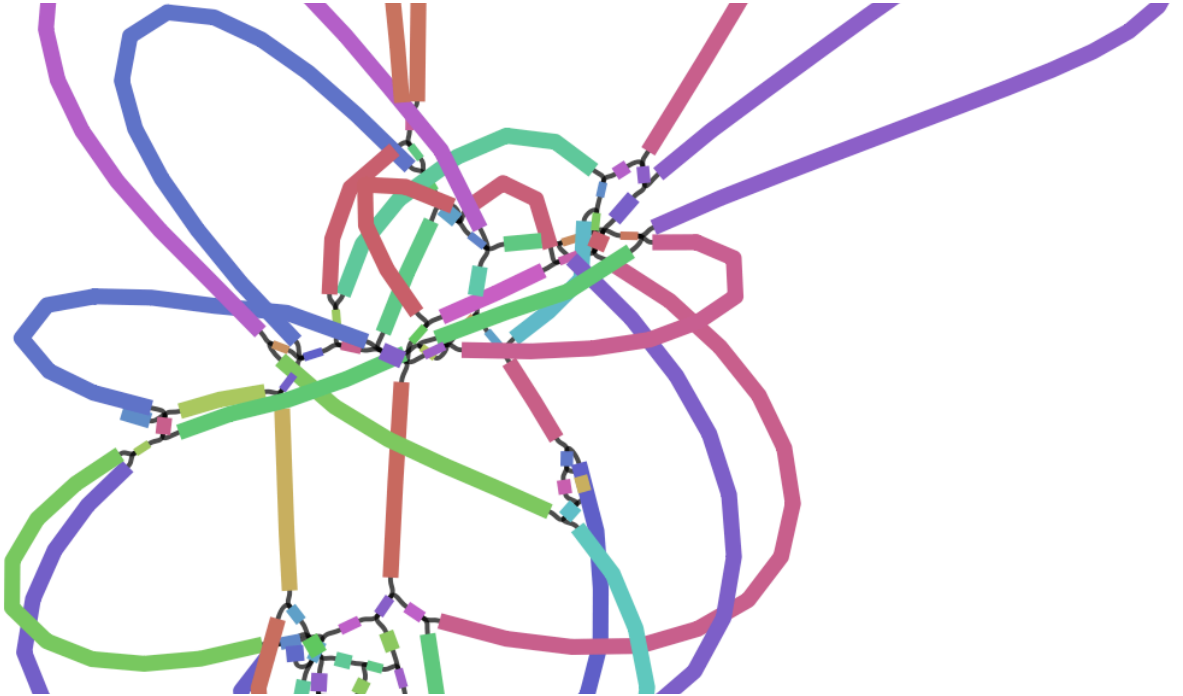
• Read length matters (ii)



• Read length matters

Read size=1000

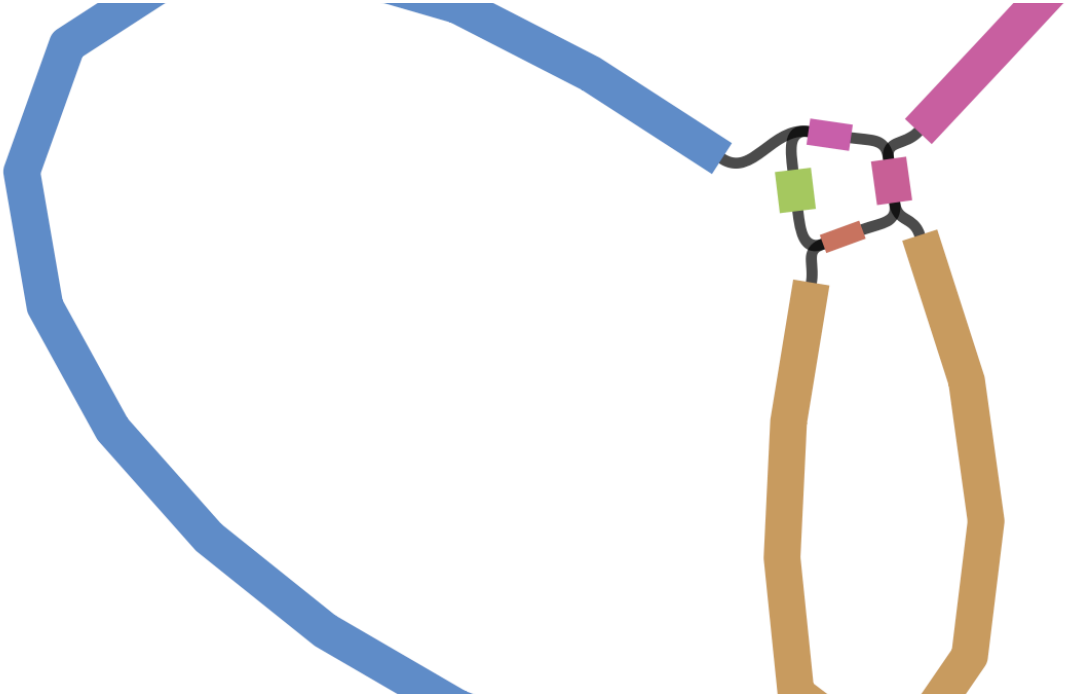
• Read length matters (ii)



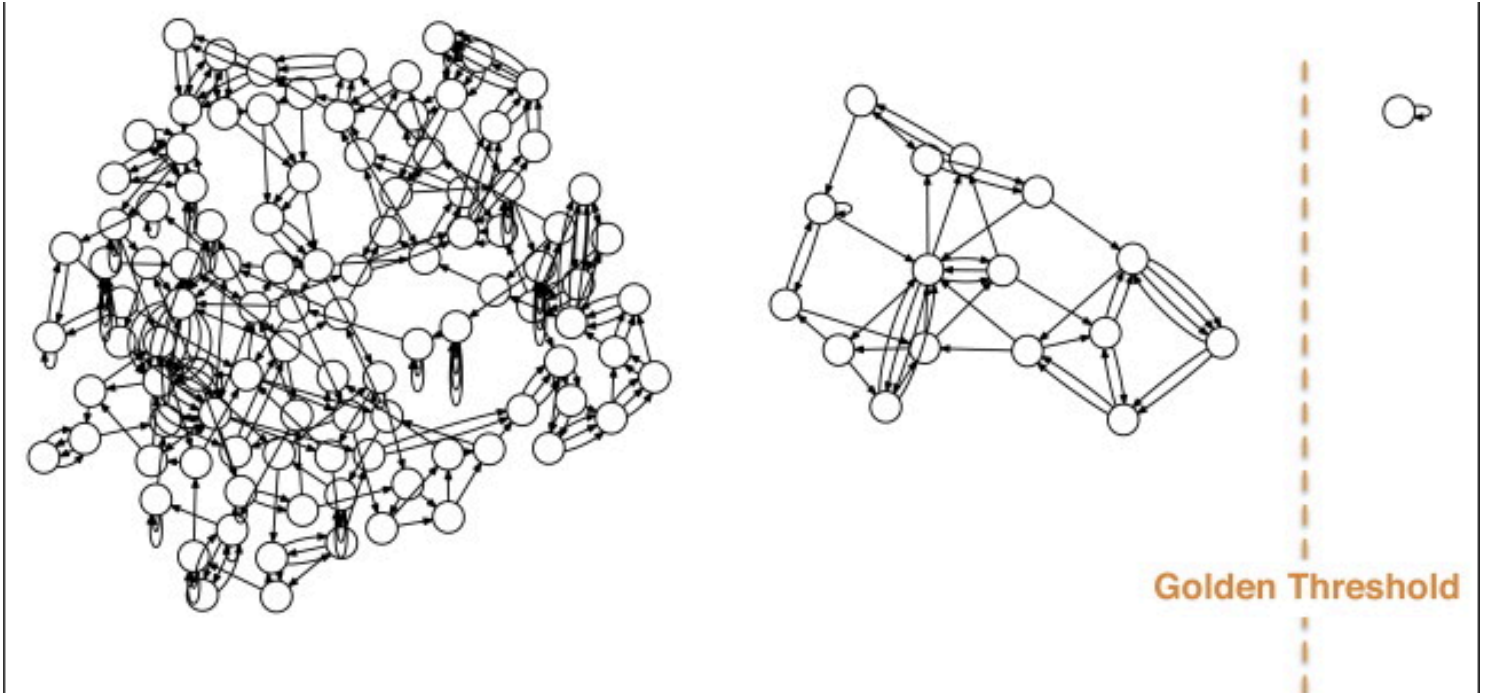
• Read length matters

Read size=2000

• Read length matters (ii)



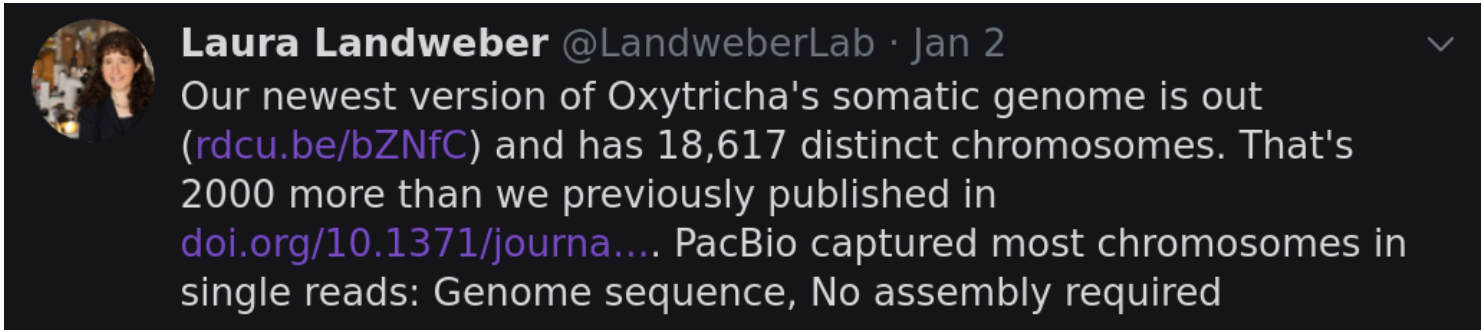
• Great hope for assembly



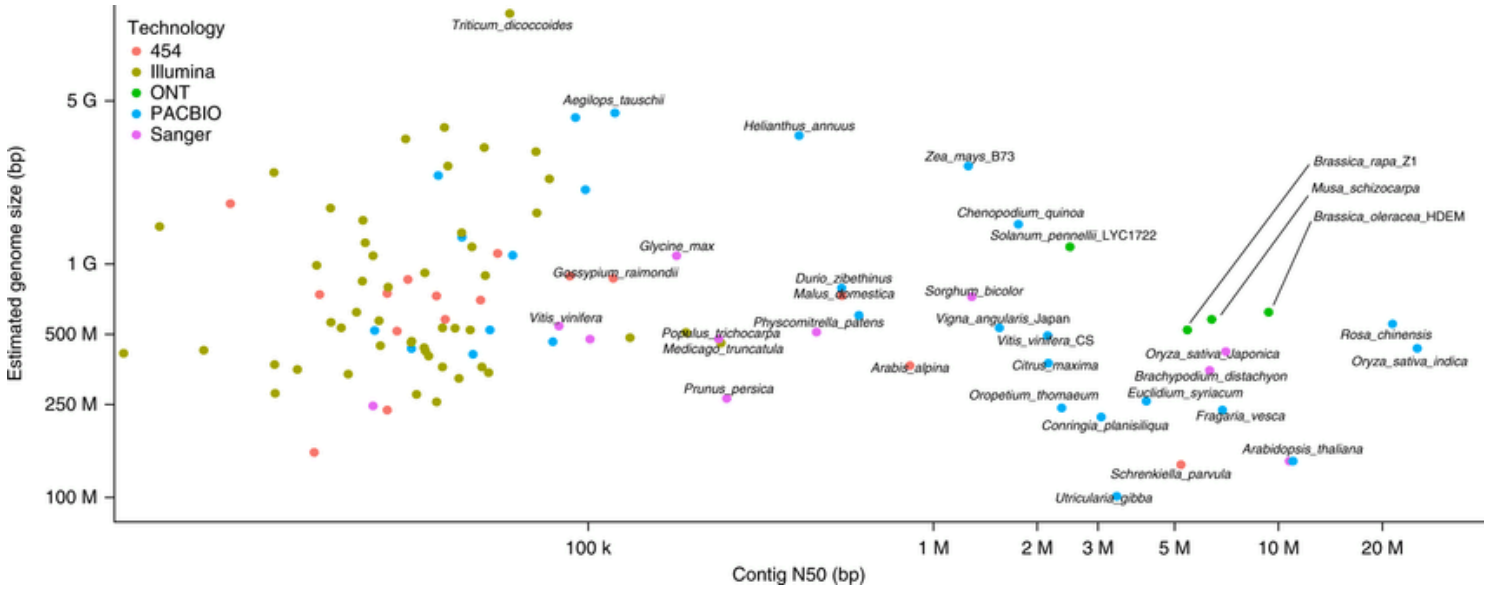
• Great hope for assembly (ii)

From “One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly” Current Opinion in Microbiology 2015

- **Long reads killed the assembly star**



• Great hope for assembly



From “Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps” Nature Plants 2018

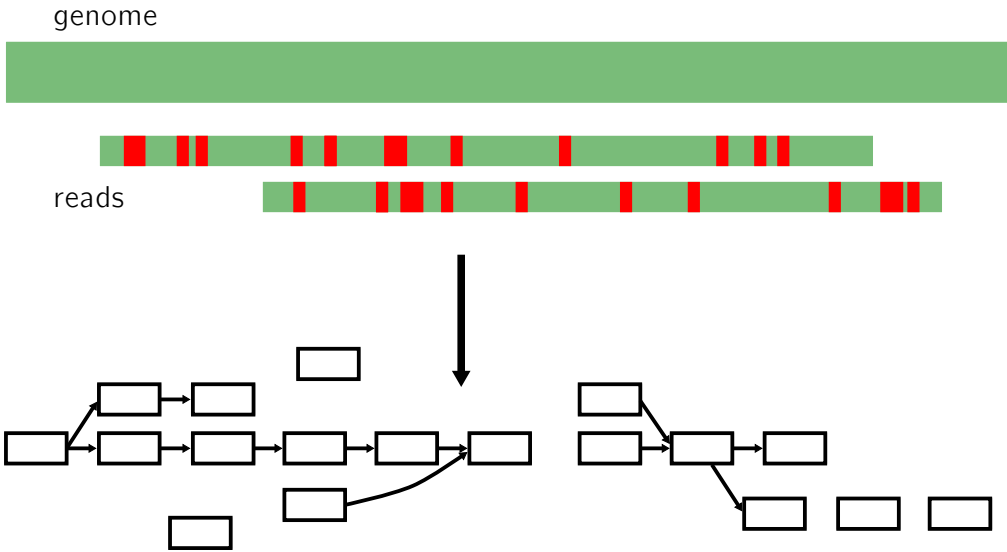
Which assembly strategy is best suited?

- Long reads approx 10 kbp
- High error rate approx 10 %
- High throughput (up to millions of reads per run) Based on long reads properties, which assembly solution would you choose and why?

Vote!

- Greedy
- Overlap graph
- de Bruijn graph

• Long reads for assembly: de Bruijn graph?



#v(0.4cm)

Most k-mers will contain at least an error and will be useless

• Long reads for assembly: overlap graph?

Supposed to be super expensive!

• Long reads for assembly: overlap graph? (ii)

```
 GAAAGCTCTG
  AAAGCTCTGA
   AAGCTCTGAA
    AGCTCTGAAT
     GCTCTGAATC
      CTCTGAATCA
       TCTGAATCAA
        CTGAATCAAC
         TGAATCAACG
          GAATCAACGG
           AATCAACGGA
            ATCAACGGAC
             TCAACGGACT
              CAACGGACTG
               AACGGACTGC
                ACGGACTGCG
                 CGGACTGCGA
                  GGACTGCGAC
                   GACTGCGACA
                    ACTGCGACAA
                     CTGCGACAAT
                      TGCGACAATA
                      …
```

Average coverage: 10
Read length: 10
Average overlap: 9
Read number:100

```
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTTCAAGTAAAAACACCTCACGAGTTAAAACACCTAAGTTC
TAAGAAAGCTCTGAATCAACGGACTGCGAC
   GAAAGCTCTGAATCAACGGACTGCGACAAT
      AGCTCTGAATCAACGGACTGCGACAATAAG
         TCTGAATCAACGGACTGCGACAATAAGTGG
            GAATCAACGGACTGCGACAATAAGTGGTGG
               TCAACGGACTGCGACAATAAGTGGTGGTAT
                  ACGGACTGCGACAATAAGTGGTGGTATCCA
```

Average coverage: 10
Read length: 30
Average overlap: 27
Read number: 33

Longer reads, better overlaps

- Less reads for the same coverage
- Larger overlaps

5Mb bacteria example with 100X coverage

Short reads

body

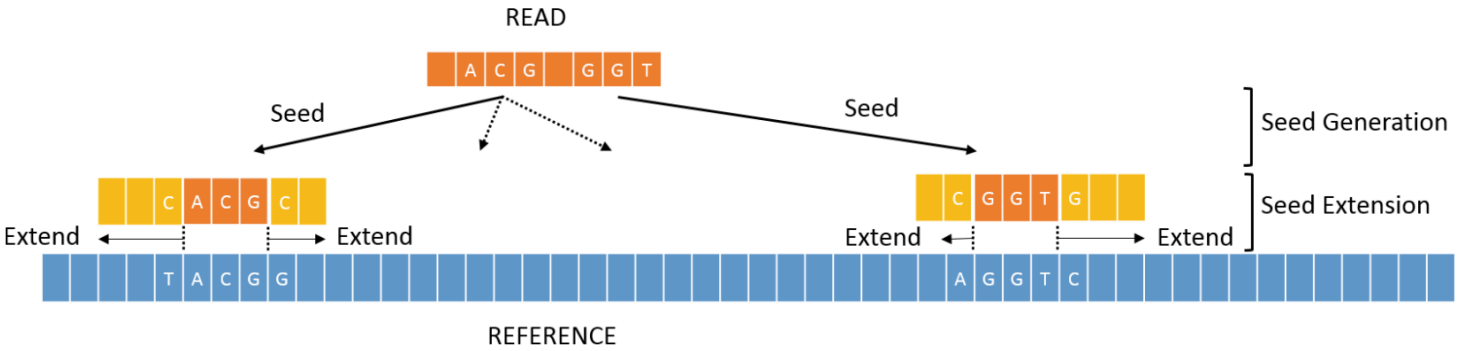
Long reads

body

Very long reads

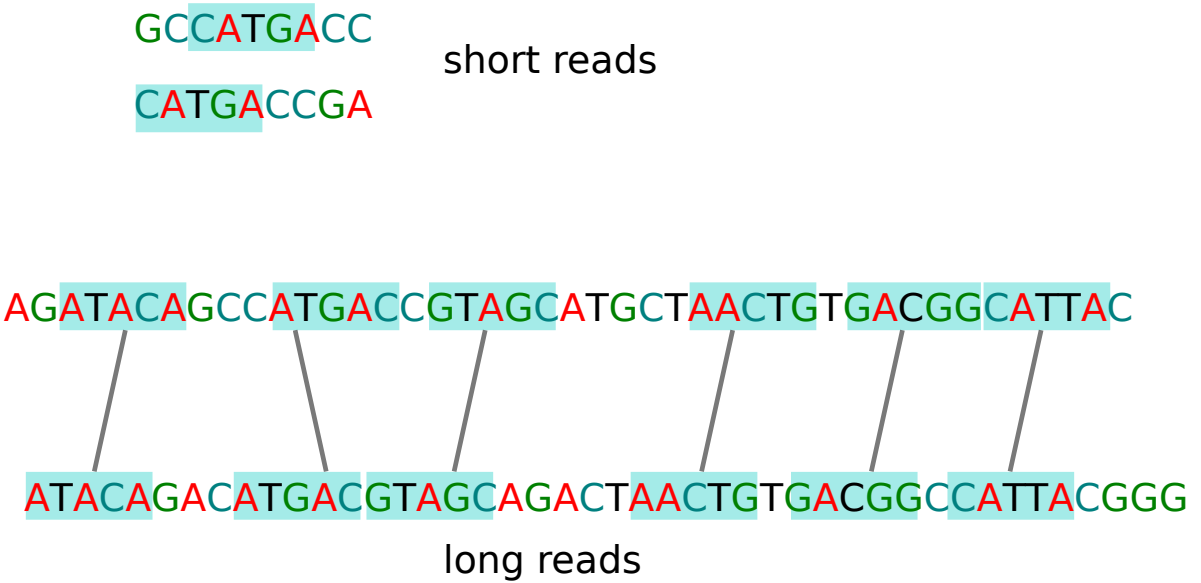
body

• Are large overlaps hard to compute?



Aligning very long and highly erroneous regions is expected to be expensive, as alignment is quadratic approx $O(n^2)$!

- **“Anchor chaining” in overlap graph**

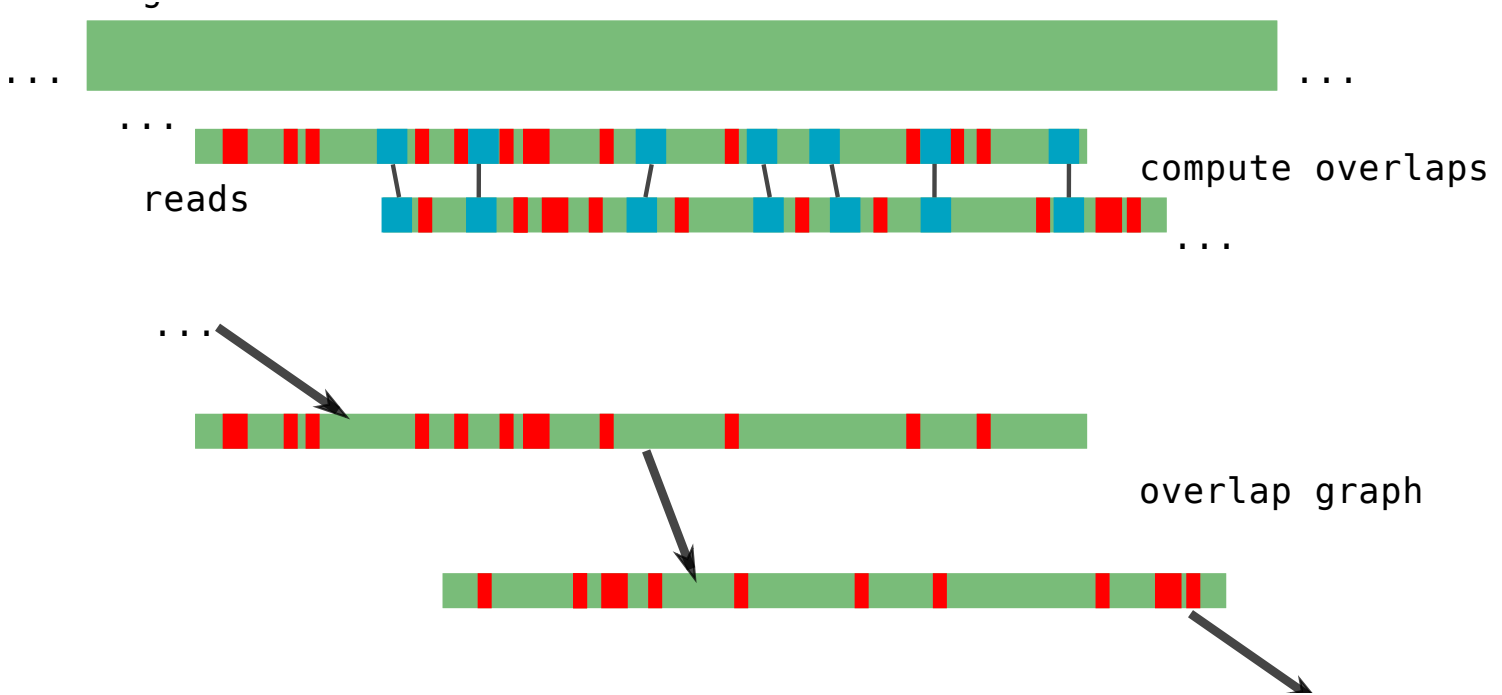


- For long reads: typically Minimap2’s [Li 2018] job

“Anchor chaining” in overlap graph (ii)

- “Anchor chaining”: find common chains of anchors (k-mers) in the same order in 2 sequences (can be **linear** in practice in most cases)

• Long reads for assembly: overlap graphs



• Sequencing errors

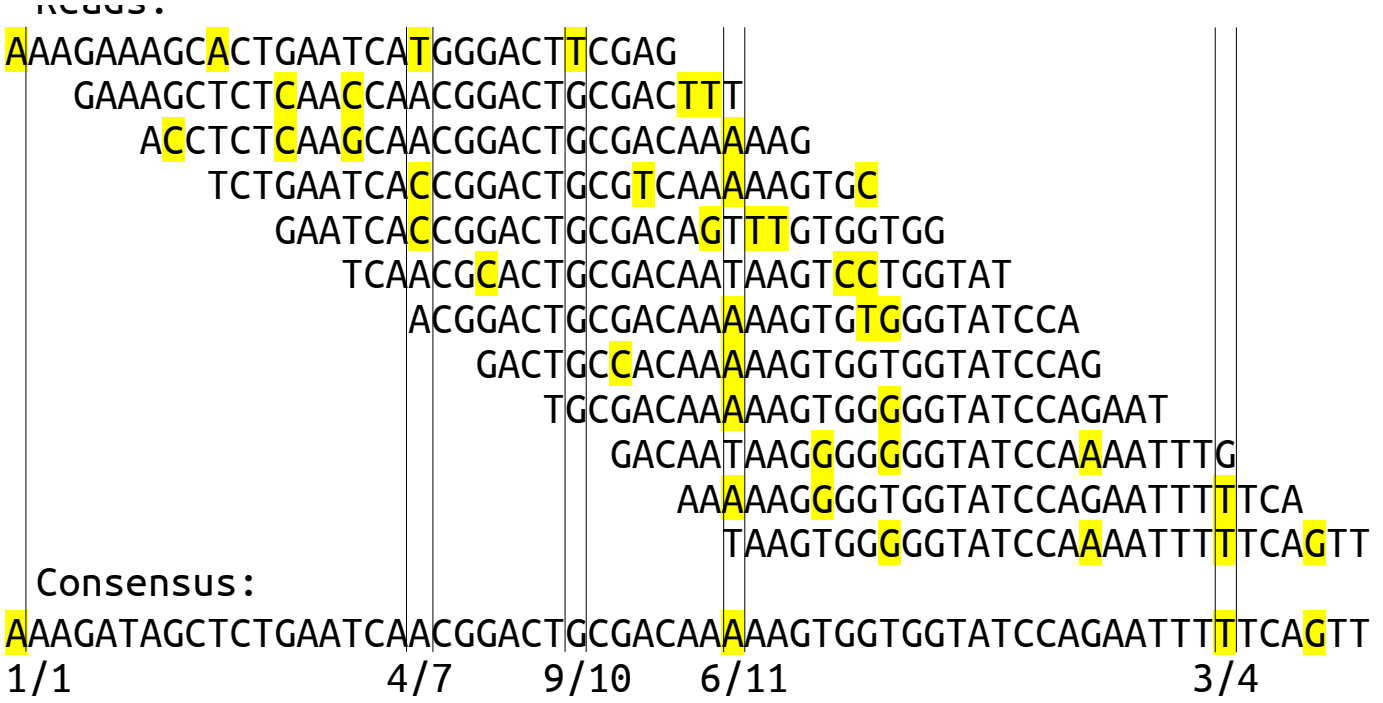
Genome:
ATCGGTATCGTTACGGTATACC

Reads:
ATCGCTATCG (Substitution)
GGTATCGTCTA (Insertion)
AT GTTACGG (Deletion)

• Sequencing errors (ii)

High rate of insertions and deletions rendered genome annotation nearly impossible

• Using coverage to remove noise: Consensus



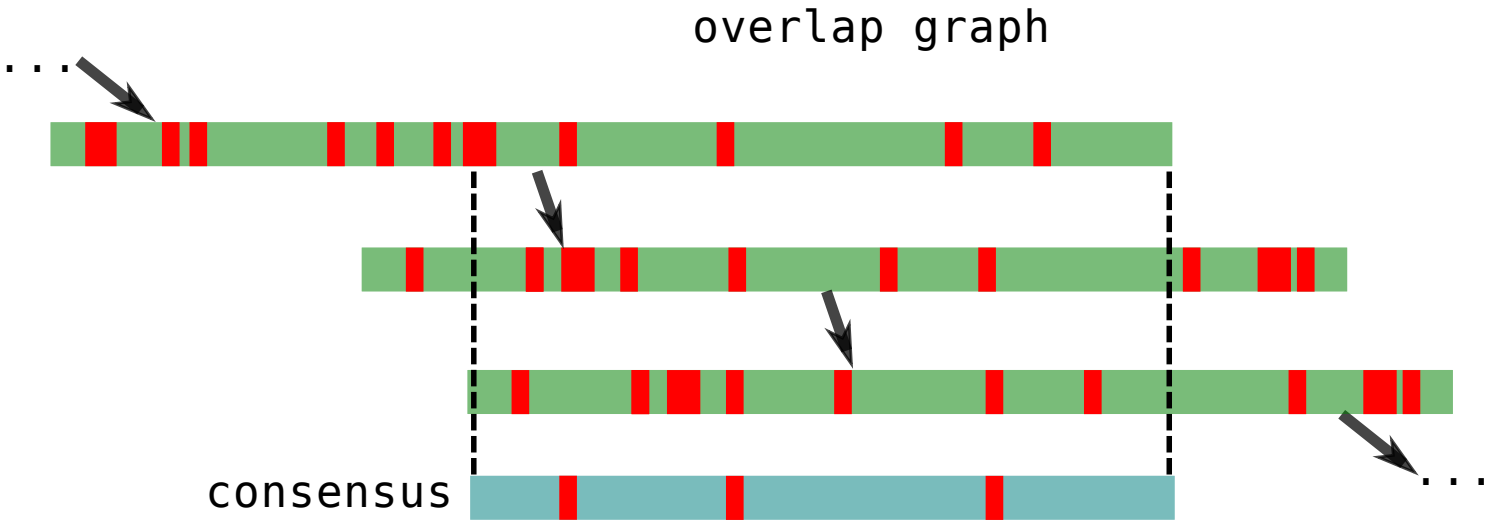
• Exercise 2: Perform a consensus

```
RS1: ACTTCGAACGT
RS2: TCGATCGTTT
RS3: GATCAGTTTAG
RS4: TCATTTCGTA
RS5: GTTTCGTCCG
REF: ACTCGAATGTTTTTCCTACG
```

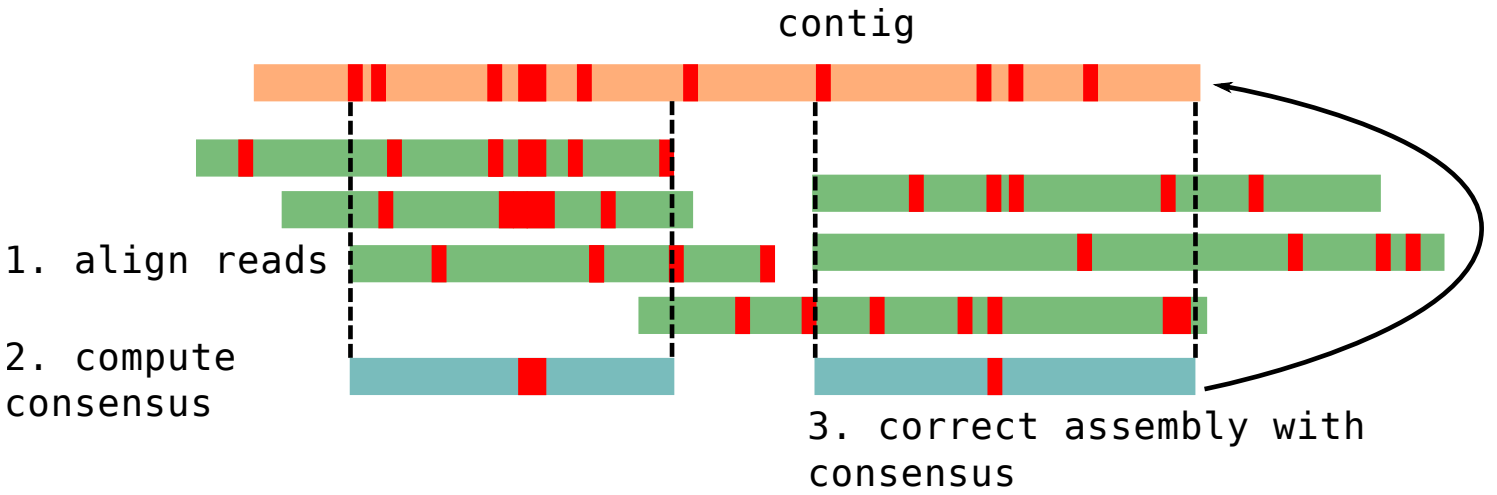
• Exercise 2: Perform a consensus - solution

```
RS1: ACTTCGA-AC-GT---------
RS2: --T-CGA-TC-GTTT-------
RS3: -----GA-TCAGTTT-AG----
RS4: ---------TC-ATTT-CGTA--
RS5: ----------GTTT-CGTCCG
REF: ACT-CGAAT--GTTTTCCTACG
CON: ACT-CGAATC-GTTT-CGTACG
```

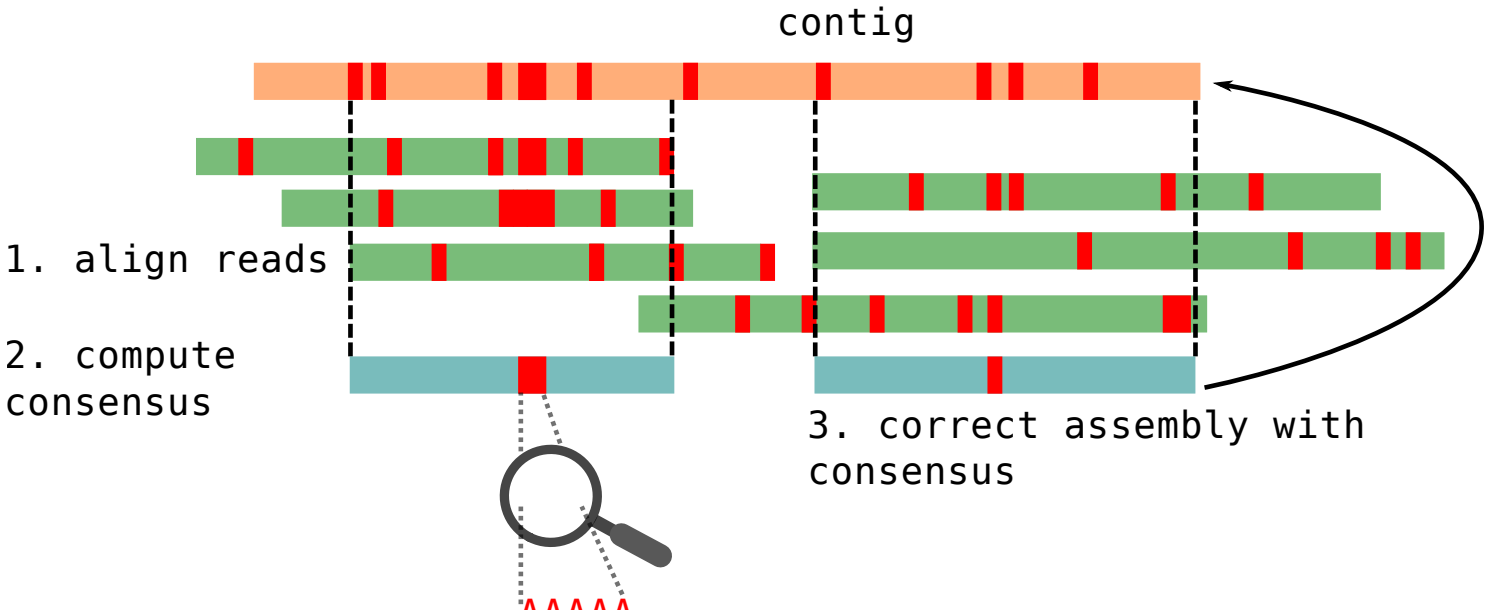
• Consensus during assembly



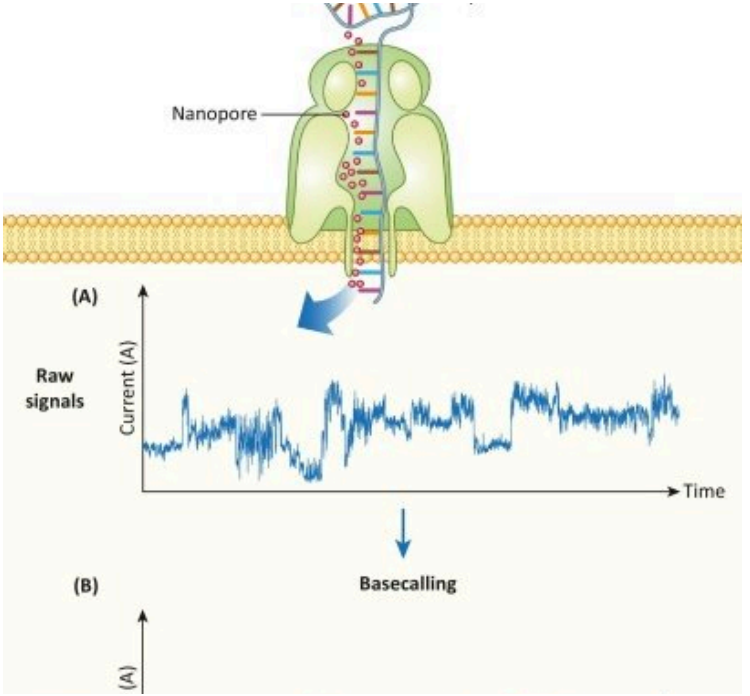
• Consensus after assembly: polishing



• Consensus after assembly: polishing



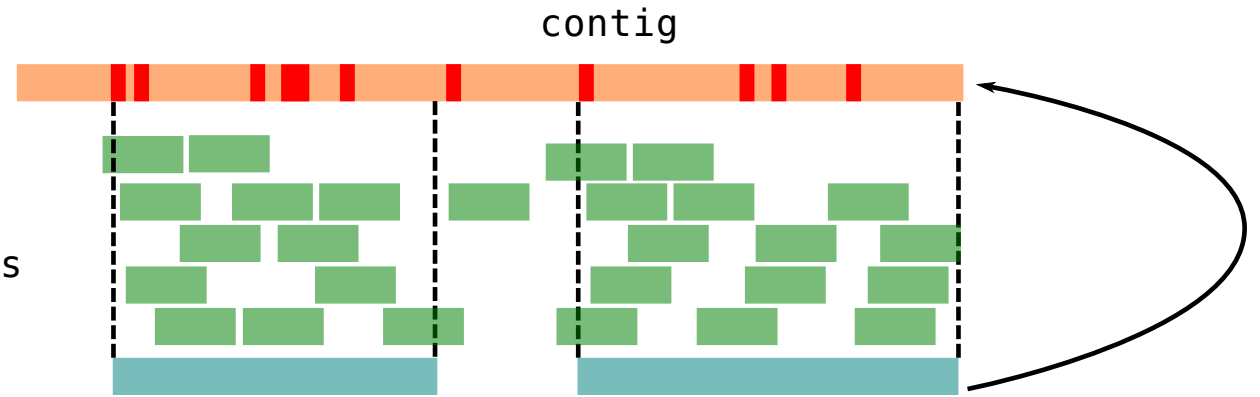
• Homopolymers are hard to read



• Polishing using accurate reads

1. align **short** reads

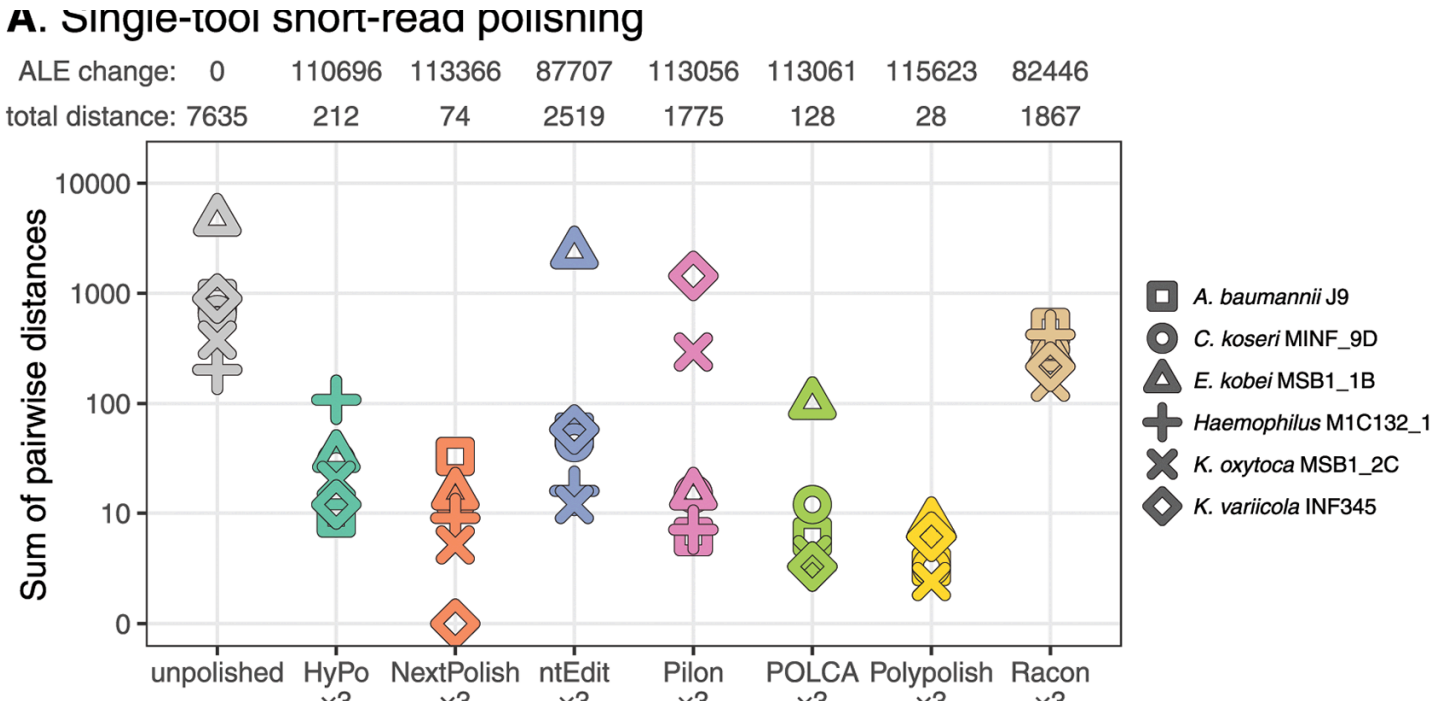
2. compute consensus



Systematics errors

Polishing with Illumina data can improve the final error rate

• Systematics errors (ii)



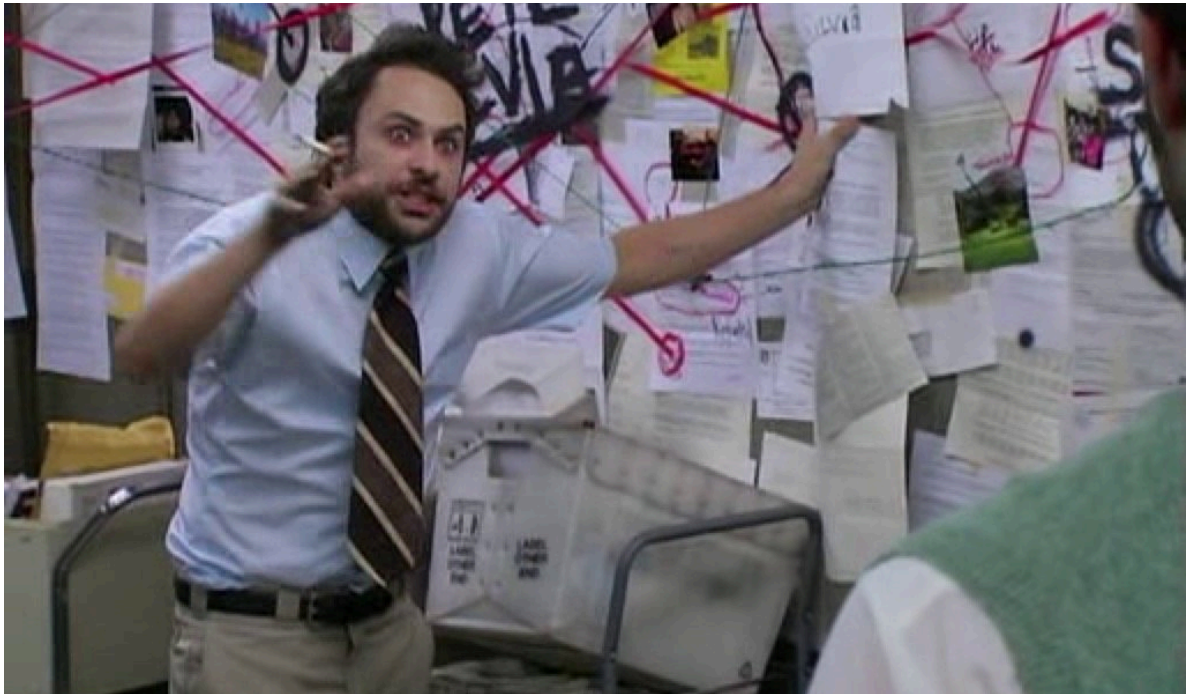
Systematics errors (iii)

From Polypolish: Short-read polishing of long-read bacterial genome assemblies

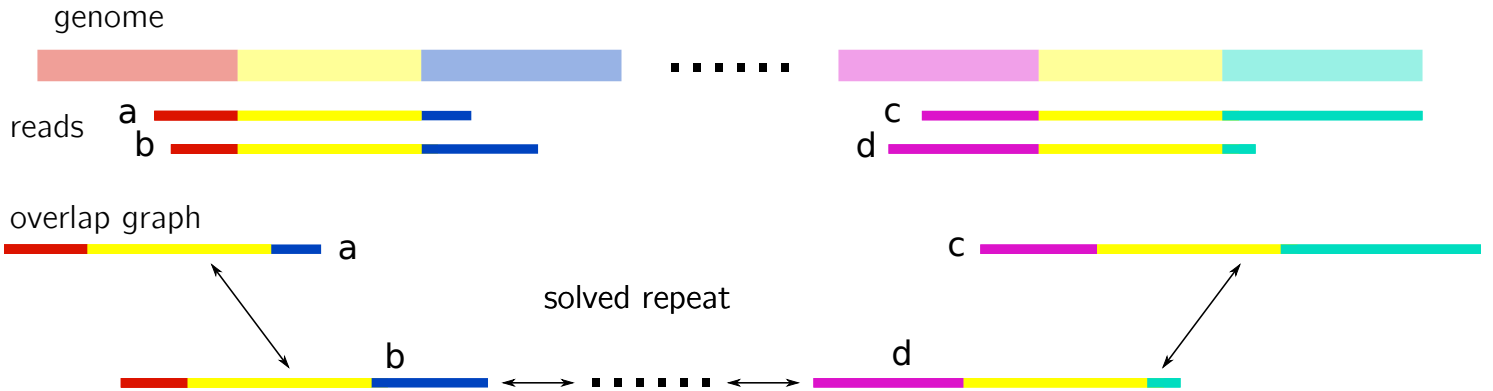
• I know we said it was the end

Just one or two more graphs

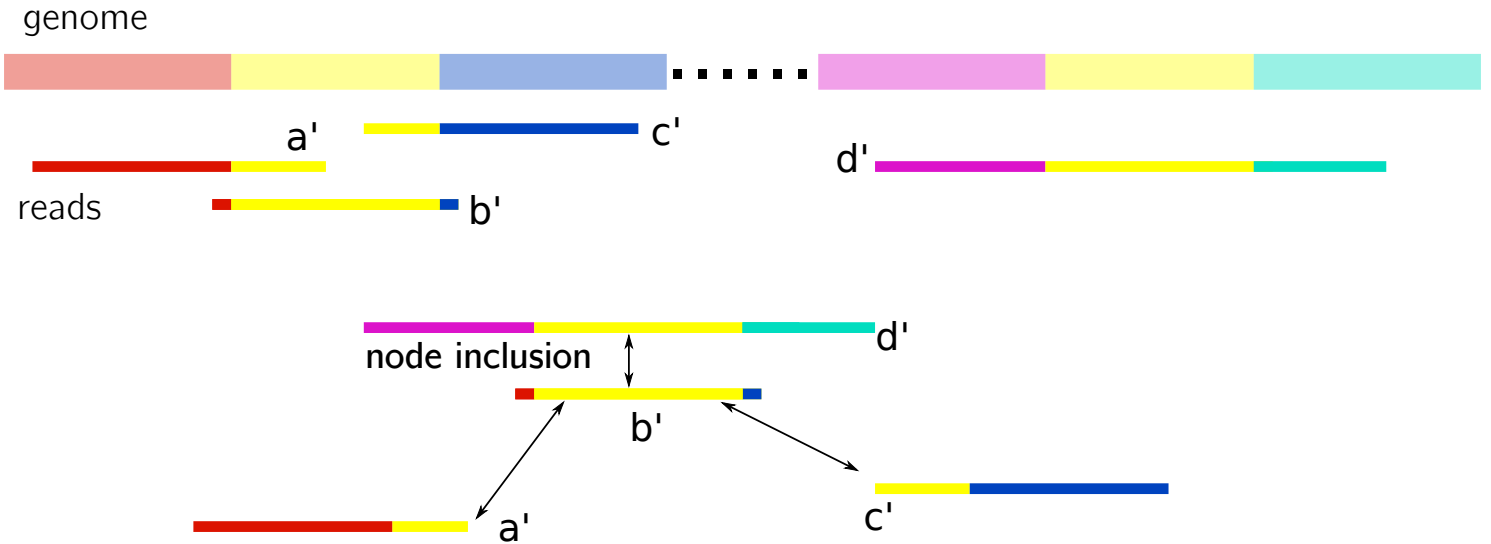
• I know we said it was the end (ii)



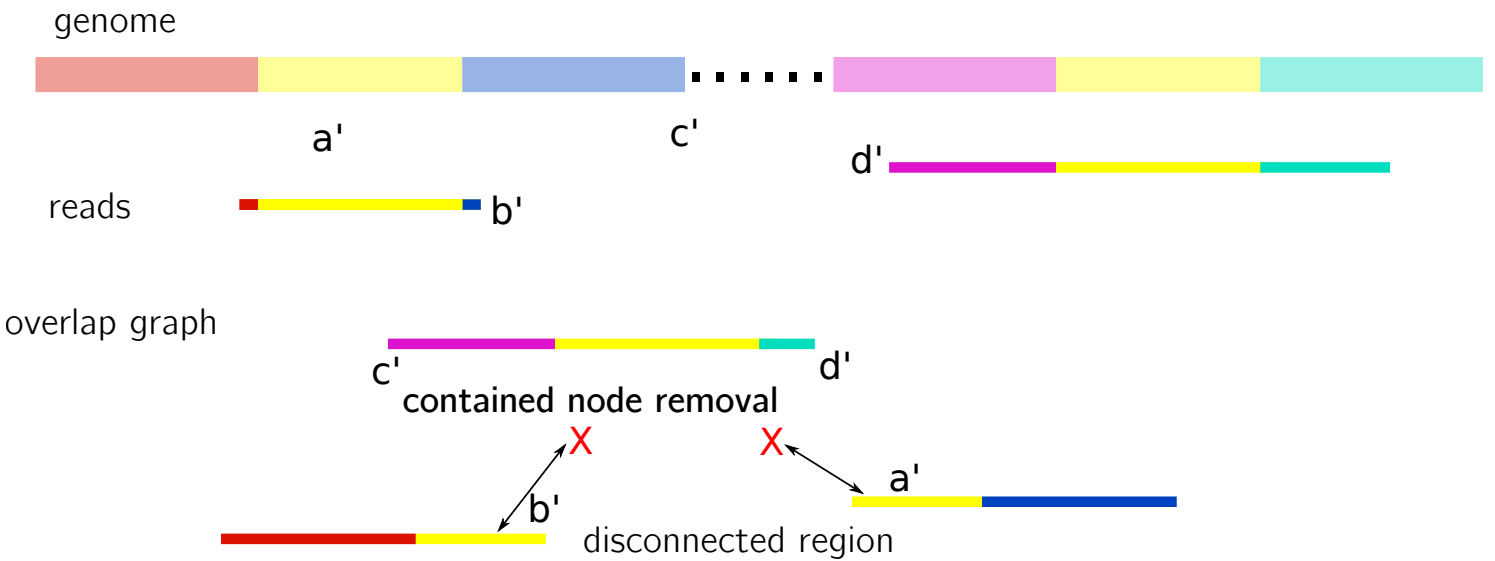
• An overlap graph limitation. Swept under the carpet?



- **An olverlap graph limitation. Swept under the carpet?**



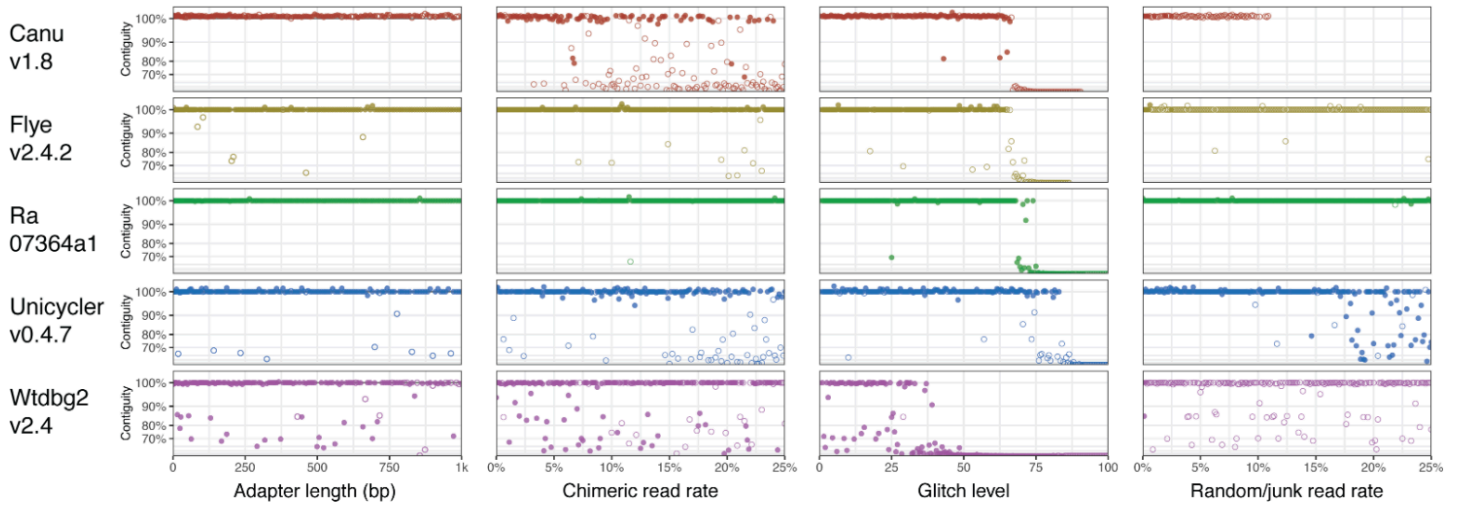
- **An overlap graph limitation. Swept under the carpet?**



• Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the software fails



From https://github.com/rrwick/Long-read-assembler-comparison

Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the data cannot solve the problem

- Very large repeated region
- Low local coverage
- Chimeric/noisy reads

• 20 years later

• 20 years later (ii)

Telomere-to-Telomere consortium

Has produced in 2021 a complete human genome with one contig per chromosomes !

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 10X Genomics, BioNano DLS and Arima Genomics HiC
- 100 authors from 50 labs

Long reads assemblers

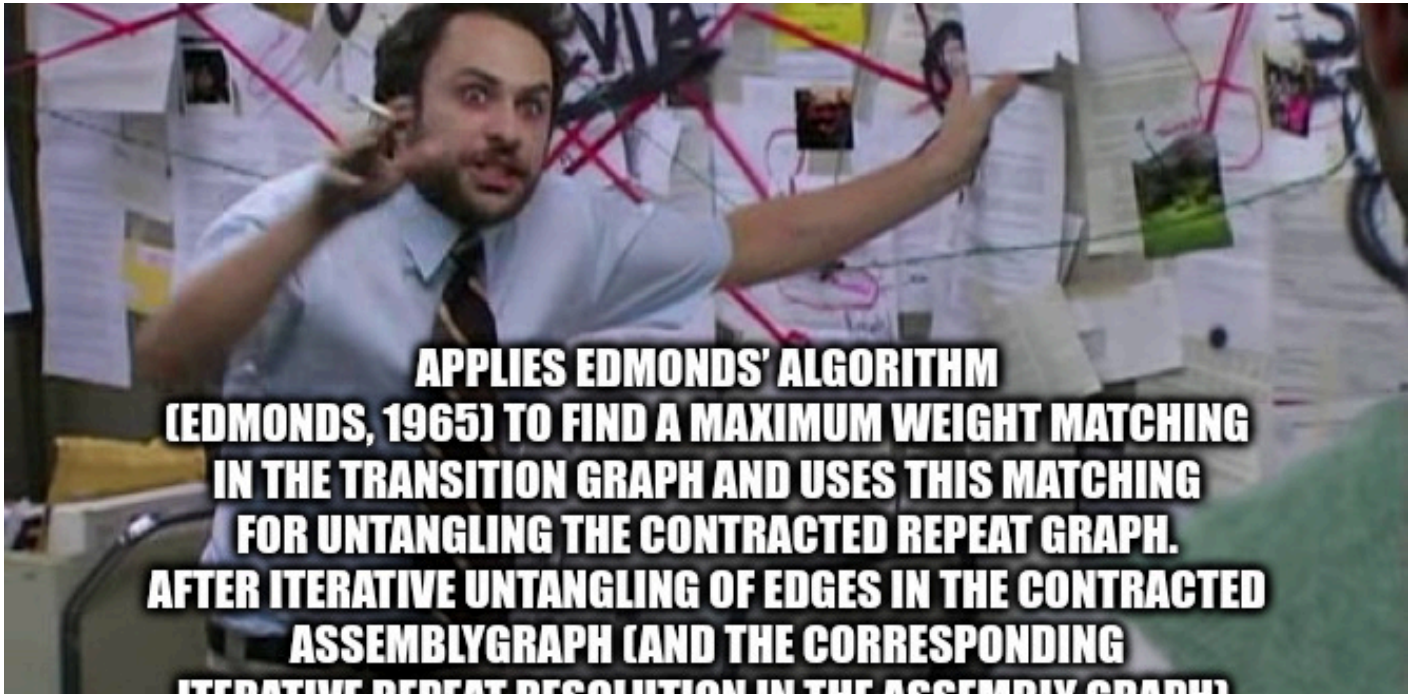
Best performing assemblers

body

Other notable assemblers

body

• Flye

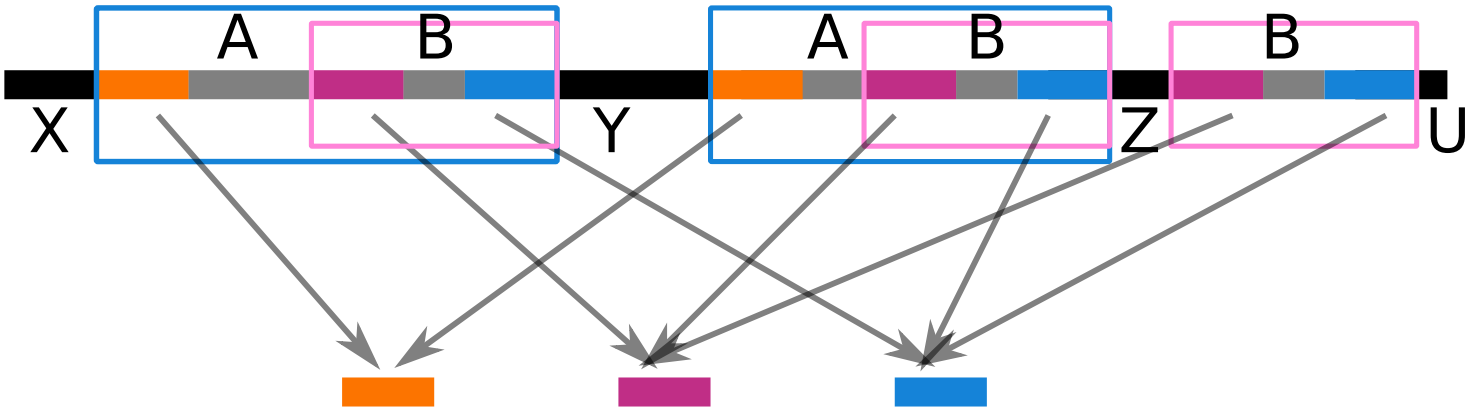

APPLIES EDMONDS' ALGORITHM
(EDMONDS, 1965) TO FIND A MAXIMUM WEIGHT MATCHING
IN THE TRANSITION GRAPH AND USES THIS MATCHING
FOR UNTANGLING THE CONTRACTED REPEAT GRAPH.
AFTER ITERATIVE UNTANGLING OF EDGES IN THE CONTRACTED
ASSEMBLYGRAPH (AND THE CORRESPONDING

• Repeat graph

a genome

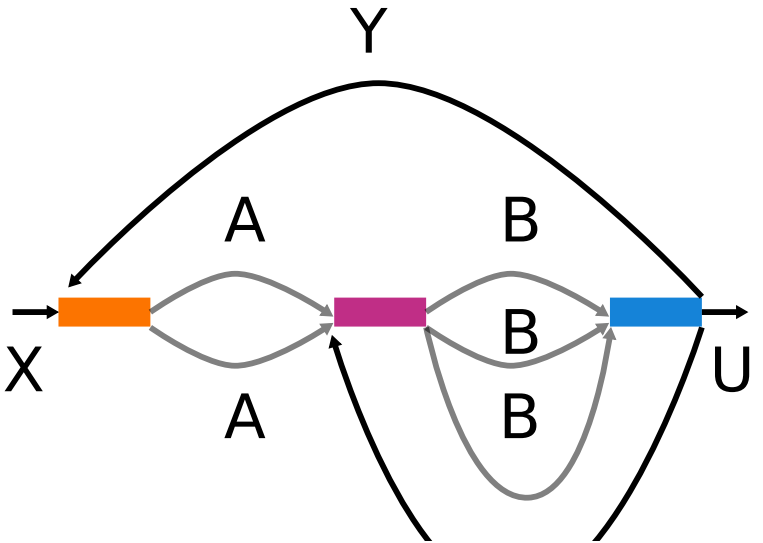
highlighted repeated regions

• Repeat graph

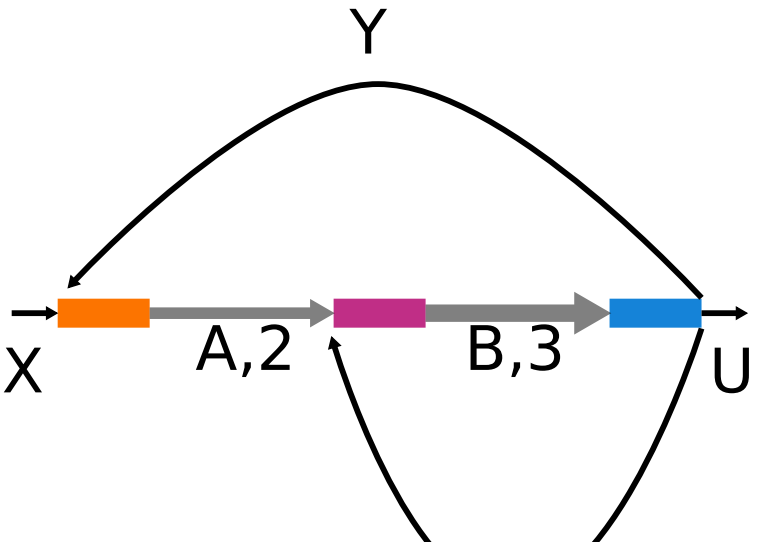


repeats extremities: graph's nodes

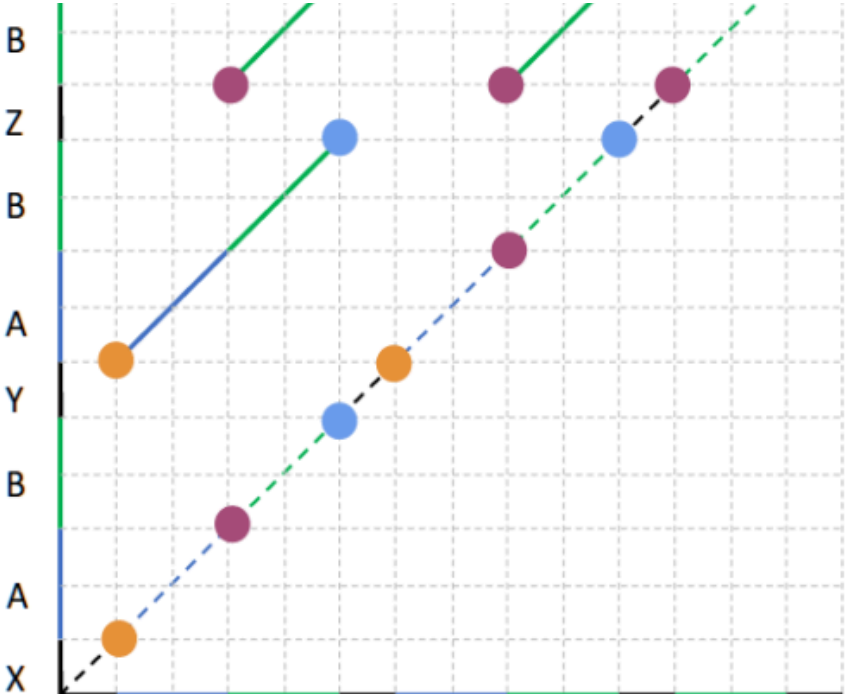
• Repeat graph

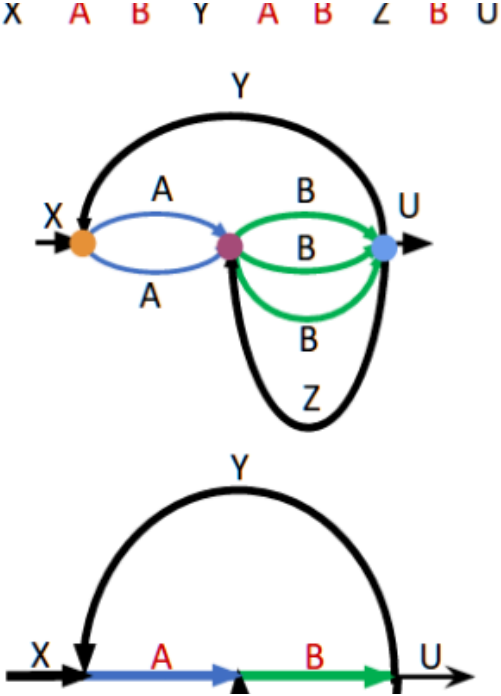


- **Repeat graph**



• Repeat graph

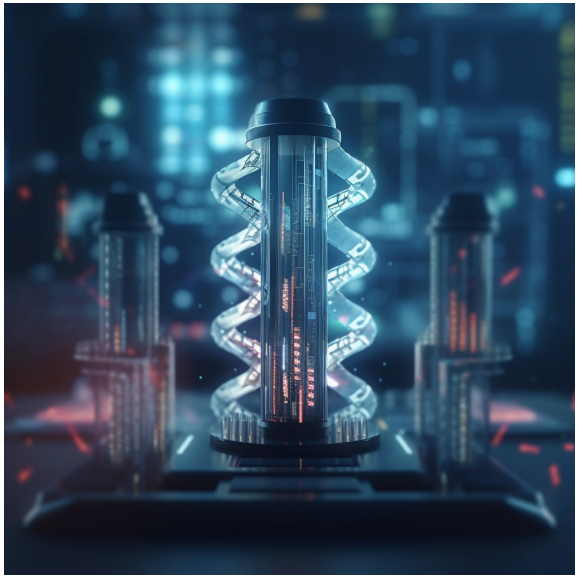




Long read assembly summary

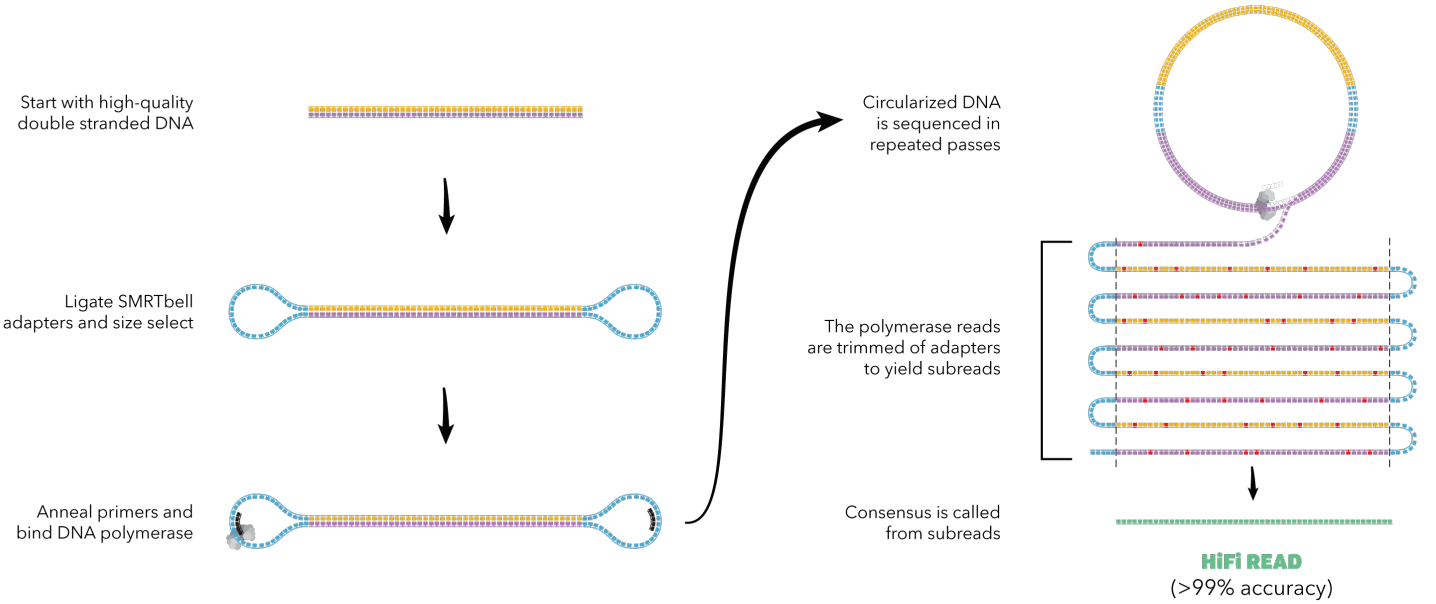
- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs

• Future of assembly



The future of genome sequencing according to MidJourney. Left: yes, but kinda boring. Right: hmmm.

• Consensus during sequencing



Consensus during sequencing (ii)

HiFi data

body

HiFi Assembly

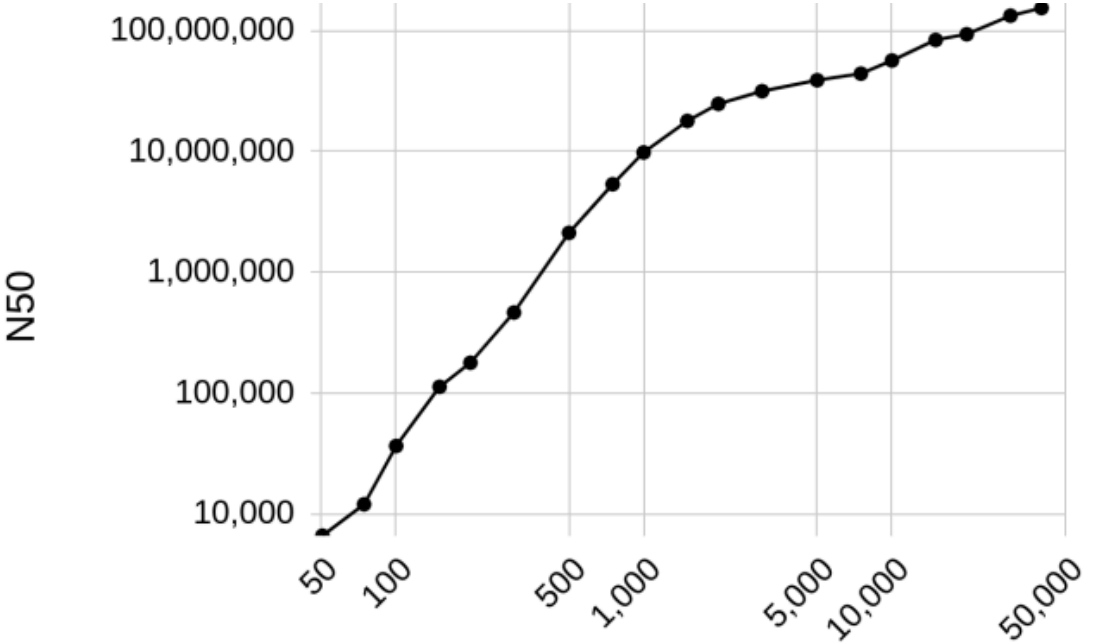
With almost error-less long reads we have several promising improvements ahead:

- Use de Bruijn graph (more efficient data structures)
- Assemble large genomes very fast
- Perform diploid assembly

de Bruijn graph Assembly

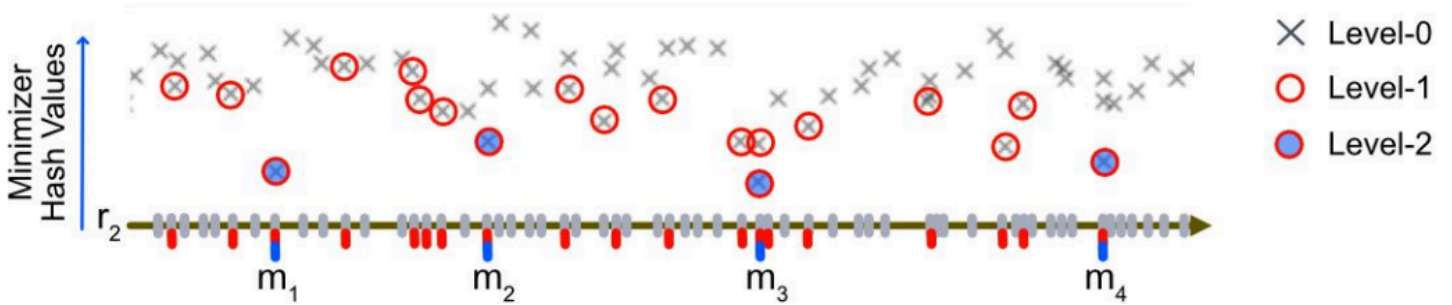
Using K=500 and K=5000 de Bruijn graphs to assemble

de Bruijn graph Assembly (ii)

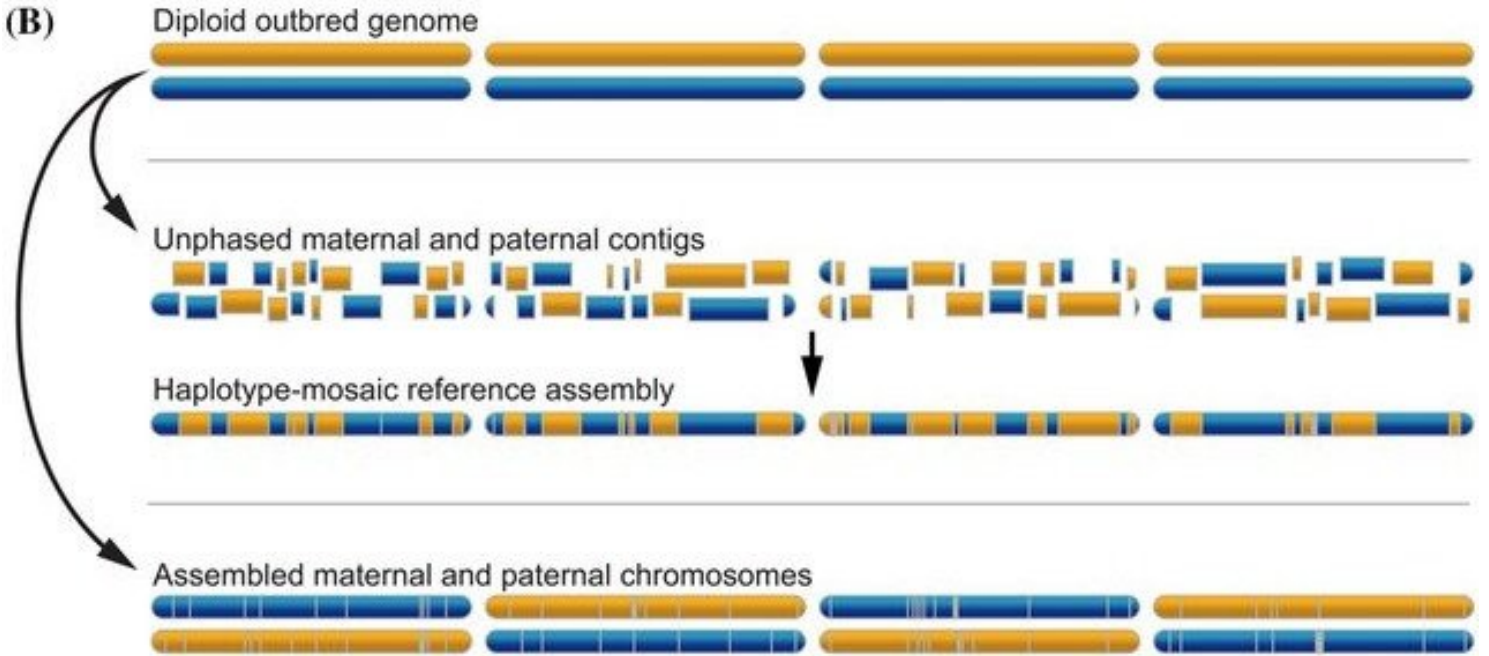


• Very fast genome assembly

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler)



• Diploid assembly



- **Human diploid assembly (20/46 chr)**

Asm	Contig NG50 (Mb)	Scaffold NG50 (Mb)	Hamming error	QV	#Errors Chr X & Y
Downsampled (35× HiFi / 60× ONT-UL)					
Verkko	12.90		0.13%	52.18	10
Verkko + trio	**80.77**	**102.55**	0.13%	52.40	10
Verkko + Hi-C	58.24	82.42	0.16%	52.48	10
LJA	0.39		0.14%	55.75	7
Hifiasm (unitigs)	0.35		0.23%	**61.05**	**2**
Hifiasm + trio	64.50		**0.06%**	60.86	26
Hifiasm + Hi-C	66.34		0.79%	60.57	37
Full-coverage (>100× HiFi / 85× ONT-UL)					

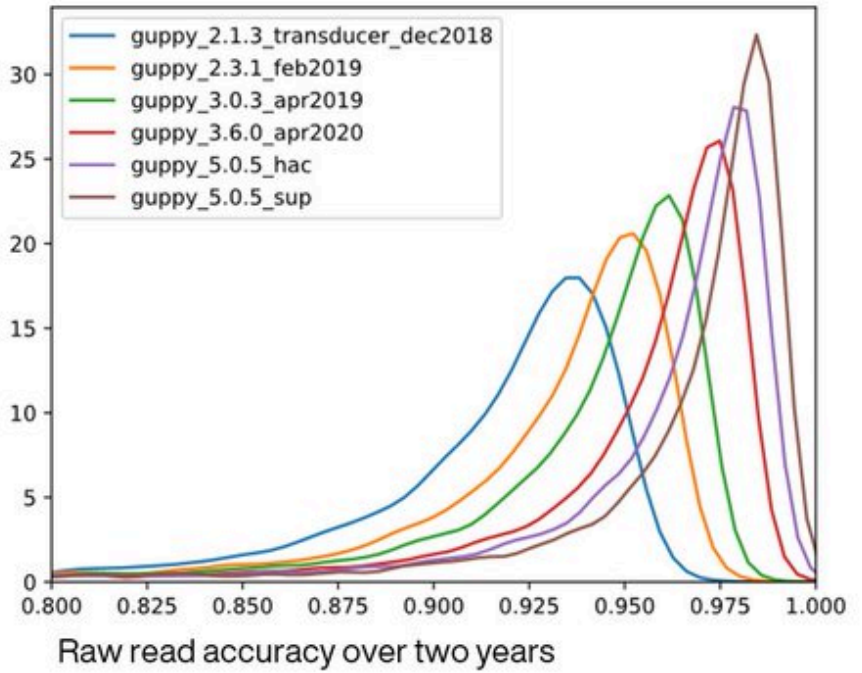
• Human diploid assembly (20/46 chr) (ii)

Verkko	17.35		**0.05%**	54.91	**5**
Verkko + trio	**134.00**	135.80	**0.05%**	55.77	**5**
Verkko + Hi-C	68.32	85.97	**0.05%**	55.57	**5**
HPRC curated	72.70	**146.75**	0.13%	**61.35**	26

• Ongoing progress

Errors in Nanopore sequencing data are rapidly diminishing

• Ongoing progress (ii)

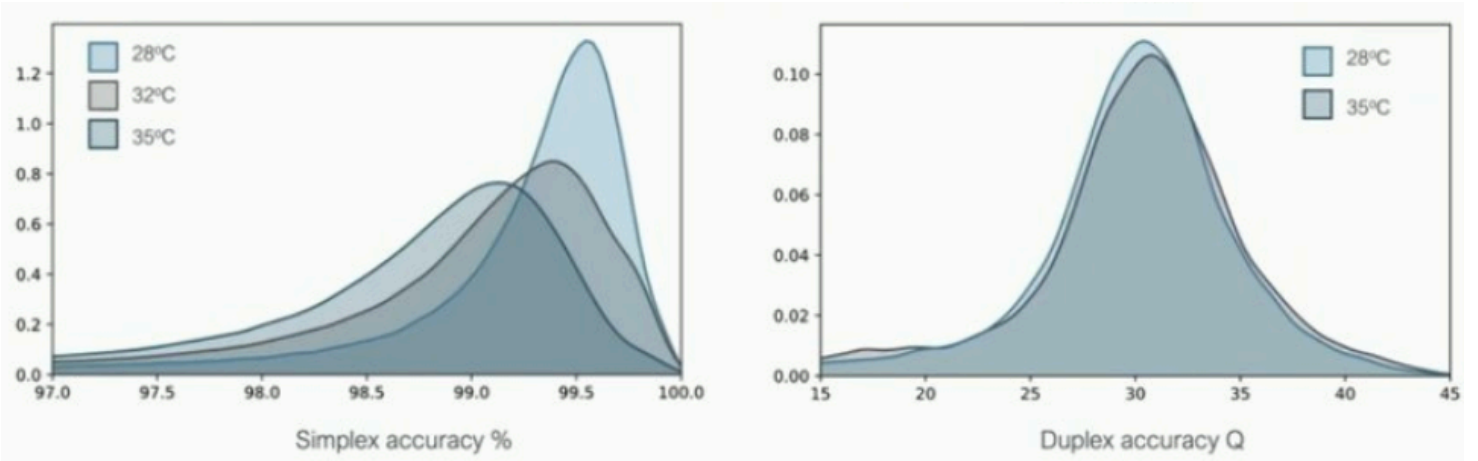


• Ongoing progress (iii)

Q20 chemistry achieved modal accuracy approx 99%

• High fidelity Nanopore incoming?

Nanopore duplex reads could deliver long and precise reads in the future



Take home messages

Ultra fast summary

body

Ongoing work

Assembly Challenges

body

The end

PopGenGoogling
@popgengoogling

i trust you to figure out your own genome

Traduire le Tweet
3:01 AM · 14 déc. 2019 · Twitter for iPhone

37 Retweets **267** J'aime